

**AUTOMATIC SENTENCE ANNOTATION FOR MORE USEFUL BUG REPORT
SUMMARIZATION**

AKALANKA GALAPPATHTHI
Master of Philosophy, University of Peradeniya, 2018

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

© Akalanka Galappaththi, 2020

AUTOMATIC SENTENCE ANNOTATION FOR MORE USEFUL BUG REPORT
SUMMARIZATION

AKALANKA GALAPPATHTHI

Date of Defence: August 14, 2020

Dr. John Anvik Thesis Supervisor	Associate Professor	Ph.D.
-------------------------------------	---------------------	-------

Dr. Yllias Chali Thesis Examination Committee Member	Professor	Ph.D.
--	-----------	-------

Dr. Wendy Osborn Thesis Examination Committee Member	Associate Professor	Ph.D.
--	---------------------	-------

Dr. John Sheriff Chair, Thesis Examination Com- mittee	Assistant Professor	Ph.D.
--	---------------------	-------

Abstract

Bug reports are a useful software artifact with software developers referring to them for various information needs. As bug reports can become long, users of bug reports may need to spend a lot of time reading them. Previous studies developed summarizers and the quality of summaries was determined based on human-created gold-standard summaries. We believe creating such summaries for evaluating summarizers is not a good practice. First, we have observed a high level of disagreement between the annotated summaries. Second, the number of annotators involved is lower than the established minimum for the creation of a stable annotated summary. Finally, the traditional fixed threshold of 25% of the bug report word count does not adequately serve the different information needs. Consequently, we developed an automatic sentence annotation method to identify content in bug report comments which allows bug report users to customize a view for their task-dependent information needs.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. John Anvik, for his guidance, support, and encouragement through my graduate experience. The input of time and comments by Prof. Yllias Chali and Dr. Wendy Osborn are very much appreciated. I extend my thanks to the members of Sibyl-lab, Palak Halvadia, Md Hasan Tareque, and Maimoona Bashir, for annotating bug report comments. I also thank the other members of the lab for their encouragement and thoughtful comments on my research work.

I would like to remind my parents who helped me emotionally and financially during stressful times. My heartiest gratitude goes to the Sri Lankan community in Lethbridge who helped me during my stay. Finally, thank you my dearest wife, Asha, for encouraging me to pursue graduate studies and for going through two hard years of a long-distance relationship.

I acknowledge the funding for this project provided by the Natural Science and Engineering Research Council of Canada (NSERC) and Alberta Innovates (Tech Futures) graduate scholarship.

Contents

Contents	v
List of Tables	vii
List of Figures	ix
1 Introduction	1
1.1 Bug Report Summarization	2
1.1.1 Bug Report Structure	2
1.1.2 Summarization Process	2
1.1.3 Prior Approaches Aim at the Wrong Target	5
1.2 Contributions of this Work	8
1.3 Organization	9
2 Background and Related Work	10
2.1 Text Summarization	10
2.2 Bug Report Summarization	11
2.2.1 Clue word score	15
2.2.2 Naïve Bayes Classifier	16
2.2.3 Support Vector Machine	18
2.2.4 Topic Modelling	19
2.2.5 Evaluating Bug Report Summaries	19
2.3 Visualization of Bug Report Summaries	20
2.4 Tagging Bug Report Comments	21
2.5 Summary	22
3 Feature Evaluation for Automatic Bug Report Summarization	23
3.1 Bug Report Features for Summarization	23
3.2 Creating an Abstract Sentence Recommender	24
3.2.1 Data Source	24
3.2.2 Extracted Features	26
3.2.3 Bug Report Summary Creation	29
3.3 Evaluation	30
3.3.1 Comparison to the Previous Approach	31
3.3.2 Evaluating Feature Groups	31
3.4 Lessons Learned	32
3.5 Summary	33

4	Automatic Sentence Annotation	34
4.1	Schemas for Bug Report Summarization	34
4.2	Automatic Labelling of Bug Report Comments	37
4.2.1	Data Set	37
4.2.2	Data Extraction and Processing	38
4.2.3	Automatic Assignment of Labels	41
4.3	A Task Relevant Bug Report Summarizer	48
4.4	Summary	50
5	Results and Discussion	51
5.1	Results	51
5.1.1	Manual Labelling of Sentences	51
5.1.2	Results of Automatic Labelling	52
5.2	Discussion	53
5.2.1	Description Labeller	53
5.2.2	Clue word Detector	54
5.2.3	Off-topic Classifier	54
5.2.4	Code Detector	55
5.2.5	Resolution Detector	56
5.2.6	URL Detector	56
5.3	Summary	56
6	Conclusion	57
6.1	Future Work	57
6.1.1	Off-Topic Labeller	57
6.1.2	Description Labeller	58
6.1.3	Plan Labeller	58
6.1.4	User Study of Customizable Bug Report Summaries	58
	Bibliography	59
	Appendix A Annotated Bug Reports	64

List of Tables

3.1	Features used for extractive summarization of bug reports	25
3.2	Comparison of our results with Rastkar’s results	31
4.1	Description of labels	41
5.1	Quality of automatic annotations	53
A.1	Annotated bug report (<i>Firefox:#495584</i>)	64
A.2	Annotated bug report (<i>Firefox:#449596</i>)	65
A.3	Annotated bug report (<i>Firefox:#491925</i>)	66
A.4	Annotated bug report (<i>Eclipse:#250125</i>)	69
A.5	Annotated bug report (<i>Eclipse:#224588</i>)	70
A.6	Annotated bug report (<i>Eclipse:#223734</i>)	72
A.7	Annotated bug report (<i>Firefox:#437797</i>)	73
A.8	Annotated bug report (<i>Eclipse:#260502</i>)	75
A.9	Annotated bug report (<i>Firefox:#328600</i>)	77
A.10	Annotated bug report (<i>GIMP:#156905</i>)	77
A.11	Annotated bug report (<i>GIMP:#164995</i>)	79
A.12	Annotated bug report (<i>GIMP:#170801</i>)	80
A.13	Annotated bug report (<i>GIMP:#364852</i>)	82
A.14	Annotated bug report (<i>GNUCash:#168803</i>)	83
A.15	Annotated bug report (<i>gvfs:#522933</i>)	85
A.16	Annotated bug report (<i>GTK:#64222</i>)	87
A.17	Annotated bug report (<i>GTK:#514396</i>)	89
A.18	Annotated bug report (<i>Eclipse:#215879</i>)	91
A.19	Annotated bug report (<i>Eclipse:#69350</i>)	92
A.20	Annotated bug report (<i>Eclipse:#226688</i>)	94
A.21	Annotated bug report (<i>Eclipse:#276131</i>)	95
A.22	Annotated bug report (<i>Bugzilla Calendar:#429126</i>)	96
A.23	Annotated bug report (<i>Thunderbird:#403907</i>)	100
A.24	Annotated bug report (<i>Thunderbird:#296655</i>)	102
A.25	Annotated bug report (<i>KDE:#153211</i>)	105
A.26	Annotated bug report (<i>KDE:#61263</i>)	106
A.27	Annotated bug report (<i>KDE:#188311</i>)	110
A.28	Annotated bug report (<i>KDE:#88340</i>)	111
A.29	Annotated bug report (<i>KDE:#66526</i>)	112
A.30	Annotated bug report (<i>GIMP:#326962</i>)	114
A.31	Annotated bug report (<i>Eclipse:#154119</i>)	116

A.32 Annotated bug report (<i>Firefox:#238215</i>)	117
A.33 Annotated bug report (<i>KDE:#173341</i>)	118
A.34 Annotated bug report (<i>KDE:#155920</i>)	120
A.35 Annotated bug report (<i>KDE:#164545</i>)	123
A.36 Annotated bug report (<i>KDE:#174533</i>)	124

List of Figures

1.1	Snapshot of the bug report of <i>Bugzilla Calendar:#1587358</i> . The Figure only shows the meta information and the first comment.	3
1.2	Snapshot of the bug report of <i>Mozilla Core:#429126</i> . The Figure only shows the description and the first few lines of the stack trace.	4
2.1	Creating a fragment quotation graph for an e-mail conversation [13].	16
2.2	Example that demonstrate how support vector machine separate two groups of data.	18
3.1	Recommender creation process.	24
3.2	Communication thread in bug report (<i>Firefox:#449596</i>)	28
3.3	Example fragment quotation graph according to the Figure 3.2	29
3.4	Average precision, recall and F-score value of leave-one-out cross validation for classifiers of different feature groups	32
4.1	Bug report summarization schema	35
4.2	Bug report sentence intention schema	35
4.3	Overview of the labeling and summarization process	41
4.4	Communication thread in bug report (<i>Firefox:#449584</i>)	43
4.5	Syntax parse tree for sentence “Feel free to open a new bug if there is something I missed.”.	47
4.6	Syntax parse tree for code <code>public static void main(String ar[])</code>	48
4.7	User interface of customizable bug report summary	49
5.1	A portion of <i>Firefox:#449596</i> bug report in Rastkar et al.’s corpus.	52

Chapter 1

Introduction

Bug reports¹ are useful software project artifacts that contain information about problems occurring in the past, discussion of possible or implemented solutions, and who contributed to these solutions. Software developers and software users use issue tracking systems to report unexpected behaviors or request new features when those are absent. Such reports and requests are stored in issue tracking systems such as Bugzilla² or Jira³. The issue tracking system records a variety of textual information including a title, description, and comments [5]. Figure 1.1 shows a bug report stored in Bugzilla for Mozilla project. Developers post comments in the bug report when they find the reported bug is in their domain of interest. The comments are a method of communication between developers that facilitate eventually solving the bug report [6]. Due to the escalation of communication, the number of sentences in a bug report can vary widely, with some bug reports containing relatively few sentences (around 20) and some having over 50 sentences.

Software developers refer to the bug report contents for various reasons. Bug report triaging [4, 5, 26, 43, 45], duplicate bug report detection [14, 23, 25, 41], and historical information collection to fix new bugs are all common software development activities involving bug reports [17, 19, 20, 22].

As the content of a bug report is primarily free-form text, natural language processing (NLP) techniques, specifically those for text summarization, are used [25, 29, 30, 31, 39].

¹We use the general term 'bug report' to refer to any software project artifact which is used for tracking project work such as feature requests, change requests, and tasks descriptions.

²<https://www.bugzilla.org> verified 03/05/2020

³<https://www.atlassian.com/software/jira> verified 03/05/2020

There are two types of text summarization techniques: abstractive and extractive. Abstractive summarization requires understanding the content of the document and paraphrasing the content in such a way that the meaning is preserved. Extractive summarization selects sentences from the document that reflect the overall meaning of the document [35]. Text summarization uses both statistical and linguistic techniques to find useful sentences in comments.

1.1 Bug Report Summarization

Bug report summaries are useful for software developers when they refer to long bug reports. Since bug reports do not have summarized versions, researchers developed various techniques to automatically summarize bug reports. In this section, we describe the bug report structure and the bug report summarization process.

1.1.1 Bug Report Structure

Bug reports contain unstructured data such as the bug report title, the bug description and comments from developers, and structured data such as platform, operating system, and version number. The unstructured data in the bug report contains full-text and semi-structured text such as stack-traces, code snippets, and URLs. Figure 1.1 shows the structured data and the bug report description that contains full-text. The full-text in some of the bug report descriptions is lengthy. In some cases, only a portion of the bug description is enough to understand the bug report. For example, the bug report #429126 of *Mozilla* has a bug report description with 151 lines. However, the bug report description ends at line 17 and the remainder is a stack trace. The first two lines in the bug description is about the platform and the version number (Figure 1.2).

1.1.2 Summarization Process

The bug report summarization studies in the literature focused on the extractive summarization approach instead of creating abstractive summaries. The reason was that abstractive

Open Bug 1587358 Opened 8 months ago Updated 2 months ago

Error when issuing "browser:purge-session-history" when browser.privatebrowsing.autostart = true

▼ Categories

Product: Core ▼
Component: DOM: Navigation ▼

Type: 🔴 defect
Priority: P2 Severity: normal

▼ Tracking

Status: UNCONFIRMED

► People (Reporter: acat, Unassigned)

► Details

Bottom ↓ Tags ▼ Timeline ▼

Alex Catarineu (Tor Browser dev) Reporter
Description • 8 months ago

Executing `Services.obs.notifyObservers(null, "browser:purge-session-history");` when `browser.privatebrowsing.autostart = true` results in a console error: `Error: _initWorker called too early! Please read the session file from disk first. SessionFile.jsm:375:15`.

This can be reproduced via UI by setting `Never remember history` in `about:preferences#privacy` and using the `Forget` (panic button) feature which can be made visible via `Customize...` in burger menu.

Jan Varga [janv]
Updated • 8 months ago

Priority: -- → P2

Release mgmt bot [cduvstro / Localize / Umarga for bugbuo]

Figure 1.1: Snapshot of the bug report of *Bugzilla Calendar:#1587358*. The Figure only shows the meta information and the first comment.

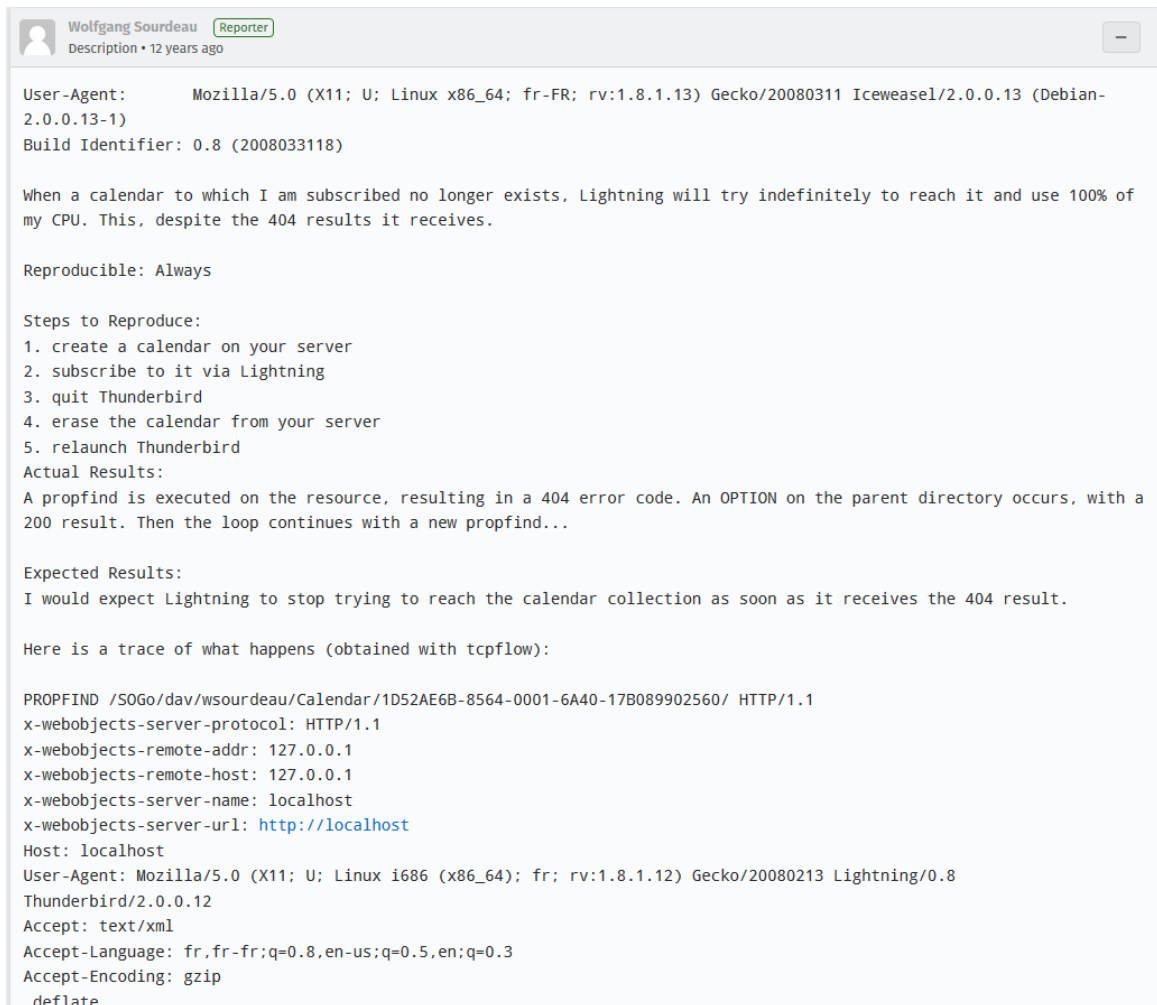
summarization has a heavy computational burden compared to extractive summarization [39] and in many cases the summaries were of poor quality [21].

Extractive summarization selects sentences from the bug report comments that are deemed to be important. Then, the selected sentences are placed in the summary in the order of how they appear in the bug report. This way the summarization model only needs to differentiate sentences (or lines) in comments that are useful or not useful to the users. We will describe extractive summarization approaches in detail in Section 2.2.

Bug report summarization studies have developed various computational techniques to select sentences for the summary. After creating a computer-generated summary⁴ the computer-generated summary was compared against one or more human-annotated summaries⁵. This evaluation method was used in many studies of bug report summarization [30, 31, 38, 39].

⁴A summary created by a computational model by selecting sentences.

⁵A summary created by a human annotator by selecting the sentences that best represent the information.



Wolfgang Sourdeau Reporter
Description • 12 years ago

User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; fr-FR; rv:1.8.1.13) Gecko/20080311 Icedove/2.0.0.13 (Debian-2.0.0.13-1)
Build Identifier: 0.8 (2008033118)

When a calendar to which I am subscribed no longer exists, Lightning will try indefinitely to reach it and use 100% of my CPU. This, despite the 404 results it receives.

Reproducible: Always

Steps to Reproduce:

1. create a calendar on your server
2. subscribe to it via Lightning
3. quit Thunderbird
4. erase the calendar from your server
5. relaunch Thunderbird

Actual Results:
A propfind is executed on the resource, resulting in a 404 error code. An OPTION on the parent directory occurs, with a 200 result. Then the loop continues with a new propfind...

Expected Results:
I would expect Lightning to stop trying to reach the calendar collection as soon as it receives the 404 result.

Here is a trace of what happens (obtained with tcpflow):

```
PROPFIND /SOG0/dav/wsourdeau/Calendar/1D52AE6B-8564-0001-6A40-17B089902560/ HTTP/1.1
x-webobjects-server-protocol: HTTP/1.1
x-webobjects-remote-addr: 127.0.0.1
x-webobjects-remote-host: 127.0.0.1
x-webobjects-server-name: localhost
x-webobjects-server-url: http://localhost
Host: localhost
User-Agent: Mozilla/5.0 (X11; U; Linux i686 (x86_64); fr; rv:1.8.1.12) Gecko/20080213 Lightning/0.8
Thunderbird/2.0.0.12
Accept: text/xml
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip
,deflate
```

Figure 1.2: Snapshot of the bug report of *Mozilla Core:#429126*. The Figure only shows the description and the first few lines of the stack trace.

The process of creating annotated summaries was different in each of the prior studies that we examined. Therefore, we present a general approach here. A human annotator creates a summary by selecting sentences that are deemed to be important to represent the bug report’s content. Since a summary should be concise, the annotator selects sentences in such a way that word count of the selected sentences falls below a given threshold. To reduce the information bias in the annotated summaries, more than one annotator is used in many of the studies. Nenkova et al. found that one should use a minimum of five annotators to have a stable annotated summary [37]. However, Rastkar et al. used three annotators [38, 39] and Mani et al. used two annotators [31]. The study conducted by Lotufo et al. used only one annotator [30]. When more than one annotator was used, the authors selected the sentence that the majority of annotators selected in their summary. Those sentences were known as the “Gold Standard Summary” (GSS) for a bug report. For example, if three annotators created summaries and at least two of them chose the same sentence, then that sentence was included in the GSS.

1.1.3 Prior Approaches Aim at the Wrong Target

Although the previous works provided significant contributions for the summarization of bug reports, we believe that they were aiming for the wrong target. In all of the prior works, the target was to create an extractive summary that was as close as possible to a GSS created by human annotators. However, based on observations in prior work, specifically those by Rastkar et al. [38, 39] and our own work [18], we have concluded that such gold-standard summaries are neither attainable in reality due to the higher word count in GSS compared to the word count threshold assigned to automatic summarizers. Also, the number of annotators used in creating annotated summaries is less than the accepted number for a stable GSS, as found by Nenkova et al. [37], which makes the evaluation less reliable. The rest of this section provides insight into how we came to this conclusion.

Creating a gold-standard is not feasible

In the previous studies, the metrics used for evaluating the summarization systems were those that are traditionally used for a recommender system: precision, recall, and F-score (Eq. 1.1, 1.2, and 1.3 respectively).

$$Precision = \frac{\# \text{ of sentences correctly selected}}{\text{total \# sentences in the summary}} \quad (1.1)$$

$$Recall = \frac{\# \text{ of sentences correctly selected}}{\text{total \# sentences in GSS summary}} \quad (1.2)$$

$$F - \text{score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (1.3)$$

Pyramid precision [37], a variation of the traditional precision metric that accounts for variation in the annotators, was also used. The annotator-created extractive summaries were used as the gold-standard when computing these metrics in these studies. However, the level of agreement, or rather the disagreement, between the annotators is a significant issue in computing these values.

Both Rastkar et al. [38, 39] and Mani et al. [31] applied the *kappa* test to measure the agreement level between annotators. Rastkar et al.’s annotators had a score of 0.41 [38, 39] and Mani et al.’s annotators had a score of 0.415 [31]. These scores indicate a moderate level of agreement between the human annotators. As Lotufo et al. [30] used only a single annotator, the validity of their annotated summaries was not confirmed.

Why does this matter? As was demonstrated by the previous studies, annotators themselves cannot agree on what is a gold-standard summary. Although pyramid precision attempts to address this issue, it has its own flaws. Specifically, this metric depends on the number of annotators (i.e. the number of tiers in the pyramid) and the reliability of the annotators. Regardless of the findings of Nenkova et al. [37], in all of the previous works,

the number of annotators was below five.

We also argue that, if the human annotators cannot agree among themselves above 50%, achieving an automatic summarization solution that has high precision or recall is not a realistic goal. More importantly, how would we know when we have developed “the best” summarization technique for bug reports? Yet, this has been the goal of previous automatic bug report summarization techniques.

According to Nenkova et al. [37] to achieve good agreement between annotators, a summarization project should use at least five annotators. However, this is not a feasible task for software engineering projects. Assume that a summarization approach focuses on producing bug report summaries for project managers. We would need five project managers to reasonably establish the gold-standard extractive summaries. Considering the resource limitations of most software projects, it is unlikely that a project would have five project managers for creating an extractive summary corpus with the needed agreement level, and project managers are only one type of bug report user. In other words, for software engineering-related summarization projects, creating an appropriately stable gold-standard corpus is not feasible.

Gold-standard summaries are unrealistic

As the primary goal of bug report summarization is to reduce the amount of time spent on reading a bug report, all prior studies focused on creating fixed-sized summaries. However, none of these approaches considered the different requirements that project members have for the summaries. In fact, it is this difference in requirements between project members that likely leads to annotator disagreement. For example, in the user study conducted by Rastkar et al. [39], they found that 58% of users would like to have reproduction steps in the summary when determining whether a bug report is a duplicate or not.

Bug reporting guidelines for projects (e.g. [1, 2, 40]) and studies on what are the important items for a bug report [7] show similar disagreements about what constitutes a "good

bug report". Such disagreements about bug report contents inevitably carry over into annotations for bug report summaries.

These points indicate that bug report summaries need to be tailored to the information needs of the user. Therefore, creating a "one-size-fits-all" gold standard corpus is an unrealistic goal.

1.2 Contributions of this Work

Although several bug report summarization approaches have been proposed [24, 25, 29, 30, 31, 38, 39], all of these studies focus on creating a fixed-size summary. However, the restrictions imposed by a fixed-size summary approach can reduce the quality of the summary by missing important or useful information for a bug report user [30, 38, 39]. Therefore, the summary of a bug report should be flexible such that the user can include or exclude information according to their information needs [10].

Previous bug report summarization efforts created condensed bug report summaries that didn't provide for user interaction [24, 25, 31, 38, 39]. Although Lotufo et al. provided two summary views: a condensed summary and an interlaced summary where the interlaced summary displayed the entire bug report by highlighting the sentences extracted as the summary [30], their approach did not allow for any user interaction with the summary. After evaluating the results and comments of user studies conducted by Rastkar et al. [39] and Lotufo et al. [30], we concluded that fixed-sized summaries that miss information sought by some bug report users is a significant flaw in these bug report summarization approaches. Therefore, we introduce an approach to automatically annotate the contents of a bug report such that an interface can allow bug report users to find and interact with the bug report to meet their task-specific information needs.

The contributions of this thesis are as follows:

- We present the results of partially replicating the study conducted by Rastkar et al. [38, 39].

- We created a bug report schema for summarization describing its content and structural dependencies.
- We created a sentence labelling schema to identify intentions of comments in bug reports.
- We developed a set of labelling modules to assign labels to sentences in bug report comments according to the labelling schema.
- We propose a tool that allows bug report users to create a customized summary for their specific information needs according to the labels assigned to the sentences.

1.3 Organization

The rest of this thesis proceeds as follows. In Chapter 2 we provide an overview of text summarization and prior work on bug report summarization. We also provide information on studies that focus on bug report visualization. Chapter 3 presents our effort to reproduce the work of Rastkar et al. [38, 39]. To overcome the shortcomings found in our replication study and the studies that we found in literature, we introduce our automatic sentence annotation process in Chapter 4. In Chapter 5, we present our results and discuss why some of the sub-modules did not perform as well as we expected. We conclude our study and outline future work in Chapter 6.

Chapter 2

Background and Related Work

This chapter provides background information on text summarization and an overview of prior work regarding bug report summarization, visualizing bug report summaries, and tagging of the contents in bug report comments.

2.1 Text Summarization

Text summarization is a trending research area because of information overload. Email threads, meeting transcripts, news articles, medical reports and software artifacts are examples for large text data where text summarization can be applied and can be useful. Text summarization has two approaches: abstractive and extractive summarization. Abstractive summarization is the process of creating a concise version by paraphrasing original content and maintaining the correct grammatical sentence structure. On the other hand, extractive summarization picks a few sentences from the original document that conveys the main idea and concatenate those sentences in an order to preserve the original meaning [36]. The following example extracted from the bug report corpus⁶ created by Rastkar et al. [38, 39] helps to distinguish between abstractive and exatractive summarization.

⁶<https://www.cs.ubc.ca/cs-research/software-practiceslab/projects/summarizing-software-aftifacts> verified 09/12/2018

<https://github.com/HuaiBeibei/IBRS-Corpus> verified 02/26/2020

Original text:

That pref was thought to be for extensions which wanted to completely replace our own Session Restore functionality. While this has worked somehow for Tab Mix Plus, we've had several issues with people ending up both Session Restore and Tab Mix Plus disabled (see bug 435055 and its duplicates). Furthermore, there are several code points which will also break when Session Restore has been disabled (such as the list of recently closed tabs). Instead of adding try-catch-blocks wherever we use Session Restore, I'd much rather encourage extensions authors to override both nsSessionStartup and nsSessionStore to provide the same API with their own functionality (or implementing a dummy-API and making sure for themselves that they've correctly replaced all known consumers)....

Abstractive summary:

It is suggested to remove the preference that removes the session storing feature, because disabling it breaks some features. It was originally intended for extensions to disable when they do their own session restore, but can leave users with it still disabled after the extension is disabled. Instead the extension writers should extend the API to do session restoration themselves. Those users concerned with the privacy implications can disable other settings to effectively to the same thing.

Extractive summary:

That pref was thought to be for extensions which wanted to completely replace our own Session Restore functionality. While this has worked somehow for Tab Mix Plus, we've had several issues with people ending up both Session Restore and Tab Mix Plus disabled (see bug 435055 and its duplicates).Furthermore, there are several code points which will also break when Session Restore has been disabled (such as the list of recently closed tabs)...

2.2 Bug Report Summarization

Over the last decade, several studies were proposed to automatically summarize software bug reports. We first summarize the three primary studies found in the literature: Rastkar et al. [38, 39], Mani et al. [31], and Lotufo et al. [30].

Rastkar et al. proposed an automatic bug report summarization model using a supervised learning technique. The authors created a bug report corpus, mentioned in Section 2.1,

with 36 bug reports extracted from four open source software projects: Mozilla⁷, KDE⁸, Eclipse⁹, and Gnome¹⁰. The corpus contains annotated summaries composed by three human annotators. Annotators were asked to create an abstractive summary at first. Then they were requested to link sentences in bug report comments that their abstractive summary was based on. Those sentences were the extractive summary of that particular bug report. The GSS were created by selecting the sentences from three extractive summaries for each bug report if at least two annotators included the sentence in their extractive summary. The GSS were used to evaluate the quality of computer generated summaries.

Rastkar et al. extracted 24 features from each sentence in a bug report and used those features to train a logistic regression model. The logistic regression model works as a classifier that selects sentences in the bug report that are suitable to add to its summary. The logistic regression model returns a value between 0 and 1 for each sentence. The authors arranged the sentences in descending order and selected the top sentences for the summary. The selection was terminated when the word count of the summary reached 25% of the word count of the bug report. The authors used the leave-one-out cross validation technique to evaluate the performance of their model. The logistic regression model received 57% for precision, 35% for recall, and 40% for F-Score. The pyramid precision for the logistic regression model was 66% [38, 39].

The study by Rastkar et al. was significant because of the bug report corpus they curated and made available to other researchers. The findings of their study motivate other researchers to pursue unsupervised classification techniques for bug report summarization. Their corpus was used as a benchmark data set when evaluating bug report summarizers in [24, 30, 31].

The next study by Mani et al. [31] took an unsupervised learning approach to create automatic bug report summaries. Similar to Rastkar et al. [38, 39], Mani et al. created their

⁷bugzilla.mozilla.org, verified 09/12/2018

⁸bugs.kde.org, verified 09/12/2018

⁹bugs.eclipse.org/bugs, verified 09/12/2018

¹⁰bugzilla.gnome.org, verified 96/12/2018

bug report corpus with 19 bug reports and hired two annotators to create both abstractive and extractive summaries. Also, they used the bug report corpus created by Rastkar et al. [38, 39] to compare the quality of the generated extractive summaries.

This study introduced an automatic “noise” identifier. Each sentence in a bug report went through three content identifiers. At first, they checked whether the sentence is a question. If not, the sentence was directed to the code identification module. If the sentence was not source code, it was checked for being an “investigative” (i.e. question) sentence. If a sentence was not flagged as any of the three types, it was labelled as “Other”. The authors found that the sentences labelled as “Other” were often greeting sentences such as “Hello” or “Thank you for your support”. The question identifier used a parsed syntactic tree structure to detect questions. The authors developed a set of rules to identify patterns in source code, stack traces, commands, and command output. A sentence was determined as “investigative” if the sentence has a minimum of five words and contains more than two keywords of the bug report. Each bug report had a keyword dictionary. Keywords were the top terms extracted from a ranked list of words whose rank was determined by the term frequency-inverse document frequency (TF-IDF) metric. The final step of the noise identifier was a filter that removes source code and “Other” sentences before passing sentences to the unsupervised summarizer.

Mani et al. [31] used four unsupervised machine learning algorithms to create summaries. Those were the centroid method (a clustering method), Maximum Marginal Relevance (MMR) [12], Graph Random-walk with Absorbing States that Hops Among Peaks for Ranking (GRASSHOPPER) [46], and Diverse Rank (DivRank) [32]. After comparing the evaluation scores, the GRASSHOPPER algorithm performed better than the other three algorithms (F-Score:50%). MMR and DivRank received scores of 48% and 46% respectively. The centroid method performed poorly with a 43% F-Score.

Like Mani et al. [31], Lotufo et al. [30] focused on creating an unsupervised bug report summarizer. In Lotufo et al.’s approach, they considered the relevance of a sentence to be

included in the summary. The relevance was measured by the similarity between a sentence and the bug report title and description, the similarity between two sentences, and the use of a heuristic to measure the agreement of two sentences. Each sentence was represented as a node in a graph. Those three heuristics were combined to calculate the weights between sentences. The weights were transformed into probabilities to convert the graph to a Markov chain. Then they applied the PageRank [11] algorithm to rank the sentence relevance.

Lotufo et al. [30] compared the evaluation scores with Rastkar et al.'s study [38, 39]. The results indicated that Lotufo et al.'s unsupervised summarisation model worked better than Rastkar et al.'s supervised learning approach. Lotufo et al.'s study stood out compared to the previous two studies because they presented the bug report summary in two manners. The condensed presentation displayed only the relevant sentences picked up by the summarizer, whereas the interlaced presentation displayed the entire bug report and highlighted the selected sentences.

There were other attempts towards creating bug report summaries. Jiang et al. applied the PageRank algorithm to rank sentences in combination with a supervised machine learning technique to assign a score to important sentences that need to be included in the summary [25]. This study also composed their own bug report corpus (called OSCAR¹¹) with 19 bug reports and 40 duplicates of those 19 bug reports. Similar to the corpus created by Rastkar et al. [38, 39], they also used three annotators to create extractive summaries. In contrast to Rastkar et al.'s study [38, 39], this summarization technique required duplicate bug reports to create better summaries. With duplicate bug reports in the corpus, their method out-performed Rastkar et al.'s [38, 39] supervised learning model. However, without the duplicate bug reports, their model's performance declined [25]. Li et al. created DeepSum, a deep learning neural network, to summarize bug reports [29]. This was the first attempt at training a deep neural network for bug report summarization. When comparing the results of their study with previous work of Rastkar et al. [38, 39], Mani et al.

¹¹<http://oscar-lab.org/paper/prst/corpus.htm> verified 08/06/2020

[31], and Lotufo et al. [30], they found that DeepSum achieved a 13.2% higher F-Score than previous studies. Since this was an unsupervised learning model, it reduced the effort of manually labelling bug reports for training purposes.

In next four sections we provided background information on feature extraction technique called the clue word score (Section 2.2.1, two text classification techniques: Naïve Bayes (Section 2.2.2 and support vector machine (Section 2.2.3, and an information retrieval method in topic modeling known as latent dirichlet allocation (Section 2.2.4 because we used those techniques in our methods.

2.2.1 Clue word score

As mentioned, NLP techniques have been used in bug report summarization because the comments in bug reports are mostly free-text. General NLP techniques, such as stemming and lemmatizing, are commonly used when preprocessing text [24, 29, 44]. Regular expressions are used to capture patterns such as code snippets and stack traces [31]. In this section, we present an uncommon techniques used in bug report summarization - clue word score (CWS).

Clue word score is one of the features extracted from bug reports in Rastkar et al.'s supervised learning model [38, 39]. CWS was introduced to summarize e-mail conversations [13]. The authors leveraged the quoted fragments in emails to find links between emails. The quoted texts were detected if the '>' symbol was at the beginning of a sentence (or line). Once the quoted fragments and corresponding original e-mails were identified, a graph was created. This graph was called the "fragment quotation graph" whose nodes were original e-mail fragments. Any unquoted text appearing before or after a quoted text were possible responses. If any unquoted text was present, those fragments were included in the graph as nodes. Then a directed edge was added from the response to the quoted text. Figure 2.1 presents the example found in Carenini et al.'s paper for a better understanding of this technique. Part (a) shows the six e-mails (E1-E6) with original and quoted texts (or

fragments) (a - j). In E2, b is a original fragment (and a possible response to fragment a) and a is a quoted fragment. Therefore, both a and b are nodes in the fragment quotation graph and an edge was drawn from b to a as shown in part (b) of Figure 2.1.

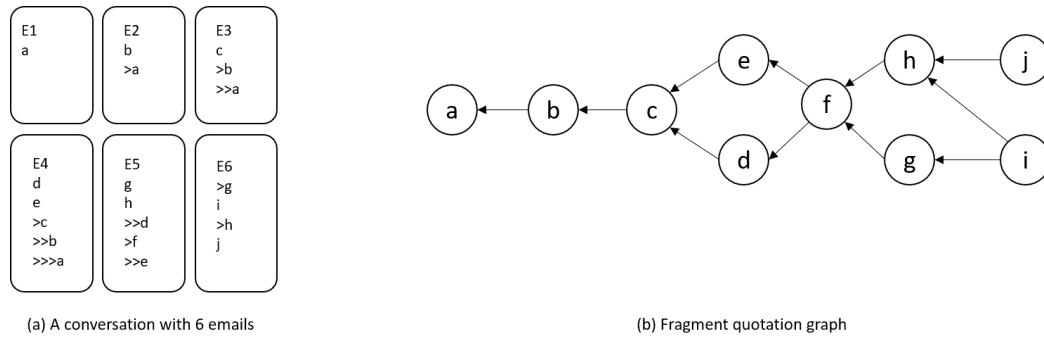


Figure 2.1: Creating a fragment quotation graph for an e-mail conversation [13].

If two words have the same meaning or same root form, these words are considered to be clue words. *Car-automobile* and *bird-birds* are considered as clue words. Clue words were counted in each node by traversing through the graph. Each node was compared with the parent and the child nodes to count how many clue words were present. The number of clue words in a fragment indicated the importance of it to the conversation.

Murray et al. tested the applicability of CWS to summarize spoken conversation [33]. But in this study, CWS was only one feature among a set of other features. Rastkar et al. applied the same technique to summarize bug reports considering the conversational nature in bug report comments [38, 39]. We have included descriptive steps of creating the fragment quotation graph and calculating the clue word score in section 3.2.2 for bug reports.

2.2.2 Naïve Bayes Classifier

The Multinomial Naïve Bayes (NB) algorithm is a widely used text classification technique. It is a generative classifier because the classification model tries to generate the most suitable class that the document belongs to based on the set of features in the document. Here the features are a set of unordered words in the documents. Since we do not consider

the order that the words appear in the document, each document is an unordered bag-of-words. Then the frequency of each word in a document is calculated. As Naïve Bayes is a probabilistic classifier, when classifying a document d to its class $c \in C$ the classifier returns the class \hat{c} which has the maximum probability. Equation 2.1 is used to calculate the probability for a given document belonging to a class.

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|d) \quad (2.1)$$

To calculate the maximum probability for a given document, for each class we find the prior probability of each class and the likelihood that a document belongs to a class. Therefore, we can rewrite the above equation as shown in Equation 2.2.

$$\hat{c} = \operatorname{argmax}_{c \in C} P(d|c)P(c) \quad (2.2)$$

The prior probability for classes can be calculated by dividing the total number of documents for a given class by the total number of documents in all classes as shown in Equation 2.3.

$$P(c) = \frac{N_c}{N_{doc}} \quad (2.3)$$

The likelihood of document d belonging to class c is calculated by dividing the frequencies of each word in a document by frequency of the same word in entire vocabulary V , then multiplying all the resulting fractions. Equation 2.4 shows how the likelihood is calculated.

$$P(d|c) = \prod_{i \in \text{positions}} P(w_i|c) \quad (2.4)$$

where,

$$P(w_i|c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)} \quad (2.5)$$

To avoid underflow and to increase the speed, NB calculations are done in *log* space.

Therefore, the Equation 2.2 can be written as follows:

$$\hat{c} = \operatorname{argmax}_{c \in C} \log P(c) + \log \sum_{i \in \text{positions}} P(w_i | c) \quad (2.6)$$

2.2.3 Support Vector Machine

Support vector machine (SVM) is a discriminative classifier that tries to separate a document from other classes based on its features. The support vector machine generates a boundary between classes that has the maximum margin between classes. The underlying mathematical concept behind training SVM is a complex process. Therefore, we will illustrate how SVM classifies data using an example (Figure 2.2).

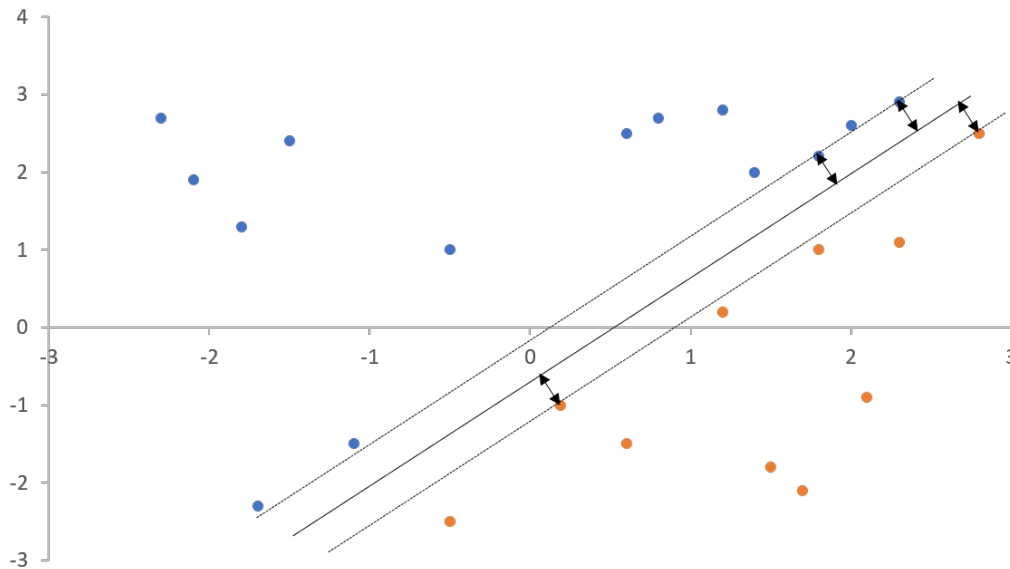


Figure 2.2: Example that demonstrate how support vector machine separate two groups of data.

In our example, we have chosen two-dimensional data so that we can represent it clearly and simply. As our data is in two-dimensional space, the boundary that separates the two classes is a line. In this case, it is a straight line. SVM defines this boundary based on the largest margin that it can create with the points from both classes. In Figure 2.2, the boundary is represented with a solid line and the margins are represented using dashed lines. As can be seen, there are two points from each class plotted on the margins, with the

boundary drawn in the middle of the margins. The idea behind SVM is that the boundary that separates the classes depends on those data points closest to it. Those points are known as support vectors because if those points shift the boundary shifts as well.

2.2.4 Topic Modelling

Topic modelling is an information retrieval (IR) method that uncovers a set of themes to which each document belongs. In topic modelling, topics are a collection of dominant words that appear in similar documents. As documents are a collection of words, the algorithm searches through documents and finds a group of words that often appear together in documents.

The Latent Dirichlet Allocation (LDA) algorithm is a commonly used algorithm for topic modelling. The documents that are passed to LDA are represented in a document-term matrix. Then this matrix is converted into two lower dimensional matrices: a document-topic matrix and a topic-term matrix where the number of topics is predetermined. Although there are initial matrices for the word distribution of topics and the topic distribution in the documents, these distributions need to be improved. Each word in the topic distribution receives a probability which is the product of two other probability measures. The proportion of word w in document d assigned to a topic t ($p_1 = P(t|d)$) is calculated and then the proportion of assignment of topic t to all documents that are derived from the particular word w is calculated from the previous probability ($p_2 = P(w|t)$). Finally, the product of p_1 and p_2 is used to update the probability of each word in the topic distributions. The process is repeated for a predefined number of iterations until the probabilities converge.

2.2.5 Evaluating Bug Report Summaries

Previous studies used precision, recall, and F-Score to evaluate computer generated summaries. As the GSSs were created using more than one human annotator, pyramid precision was also used to evaluate summaries. The pyramid precision metric was introduced

by Nenkova et al. to measure the relative importance of information included in a summary [37]. To calculate Pyramid precision, first, we count the number of times each sentence in a summary was picked by an annotator. Then this number was divided by the number of sentences with the highest possible links in the summary as shown in equation 2.7.

$$\text{pyramid precision} = \frac{\sum \# \text{ links for a sentence in a summary}}{\text{highest possible links for the same \# sentences}} \quad (2.7)$$

For example, if a summary was generated with five sentences and three of the sentences were linked by two annotators and remaining two sentences were linked by a single annotator, the total number of links in the summary is $(3 \times 2) + (2 \times 1) = 8$. If the annotated summaries for the same document (or a bug report) has four sentences with three links and five sentences with two links and rest of sentences were only linked by one annotator, the best possible number of links for the same size summary is $(4 \times 3) + (1 \times 2) = 14$. Therefore the pyramid precision for the document is calculated as $\text{PyramidPrecision} = \frac{8}{14} \approx 0.57$.

2.3 Visualization of Bug Report Summaries

The Rastkar et al. [38, 39] and Mani et al. [31] studies did not address the visualization of the bug report summaries. However, Lotufo et al. created two different views for a summary; one that only displays the summary sentences and the other that has the entire bug report and highlights the summary sentences [30]. The user study conducted by Lotufo et al. found that 56% preferred the condensed summary view whereas 46% preferred the interlaced view [30]. The comments from that user study indicated that some users preferred the interlaced view as they did not trust the automatic summarizer. However, the sentence highlighting allowed such users to read through the comments quickly. On the other hand, users did not find that the interlaced view is useful with very long bug reports.

A study that did focus on bug report summary visualization was by Yeasmin et al. [44]. Their visualization approach is similar to Lotufo et al.'s technique [30]. However, instead of using one colour, Yeasmin et al. used different colours for the summary sentences so that

a user can focus his/her attention on the specific colour coded sentences in the original bug report.

2.4 Tagging Bug Report Comments

As part of their work on providing an efficient interface for bug report triaging, Bortis and van der Hoek commented that information in bug report summaries should be interpretable at-a-glance but still have the descriptive information readily available [10]. In their work, tags were used to mark bug reports with common characteristics. Similarly, in our work, we developed tags to mark sentences with common attributes such as code snippets, URLs, and quotes from other comments.

Mani et al. found that identifying and removing noise when training the classifier improved the performance of the classification model. They identified syntax, stack traces, and error messages (labelled as *code*) and greeting sentences (labelled as *other*) as noise [31]. Similar to Mani et al.'s approach we identify but do not emphasize the sentences that have noise tags (e.g. URL, code, and off-topic). We provide an interface that presents information to understand the bug report with the flexibility of viewing other information if desired. Therefore, unlike Mani et al.'s approach which completely removed noise sentences for the user, we allow selective access to any information in the bug report.

Rastkar et al.'s corpus contains tags that describe the intent of a sentence. They used the labels *problem*, *suggestion*, *fix*, *agreement*, *disagreement*, and *meta* [39]. The *meta* label was used for code snippets, stack traces, and error messages with the other tags being self-explanatory. Huai et al. also used the idea of intentions to improve the classification model [24]. In our work, we chose to refrain from using those tags because of the inherent problems with human annotated information. Instead, we proposed a broad set of tags that can be extracted based on keywords (or key phrases) and regular patterns such as URLs or enumerated lists.

2.5 Summary

Previous studies have created supervised and unsupervised bug report summarization models. Unsupervised models performed well as summarizers when generating the summaries for new bug reports, whereas supervised models were sensitive to the training data, and therefore did not perform as well. NLP techniques were widely used in summarization studies when extracting features and finding structural dependencies. Annotated summaries were used to evaluate automatically generated bug report summaries. Precision and recall were widely used metrics when evaluating summaries. When more than one annotated summaries are available, pyramid precision also can be applied as an evaluation metric. Very few studies focused on bug report visualization while the majority of the studies simply displayed the extracted sentences from a bug report as a summary.

Chapter 3

Feature Evaluation for Automatic Bug Report Summarization

Rastkar et al. [38, 39] presented an extractive approach to bug report summarization. Their approach selected sentences to be extracted from the bug report based on a variety of features extracted from sentences. As part of our work on bug report summarization, we investigated whether reasonable extractive summaries of bug reports can be created without the use of the complex features used on Rastkar et al.’s approach [38, 39], and how the different features contribute to the creation of a reasonable extractive summary. The linear regression model trained by Rastkar et al. achieved 57% precision and 35% recall. We used the word “reasonable” in here to emphasize how useful the sentences extracted from bug reports to create a summary for a software engineering task considering the moderate scores achieved by the classifier.

3.1 Bug Report Features for Summarization

Rastkar et al. used 24 features that were categorized into four sets: sentence length, lexical features, structural features and features related to the participants of the conversation in the bug report [38, 39]. As Rastkar et al. found that the F-score statistics of the structural and participant-related features have low variability, we focused our study on the length and lexical features (shown in bold in Table 3.1) [38, 39]. Also, we chose to remove the complex features, those related to sentence entropy, to investigate how extractive summaries are selected without the use of these features and compare them to manually

created summaries. In short, we investigated the use of two sentence length features, six conditional probability scores, four cosine similarity scores, and the clue-word score. The details for these features are explained in the following two sections.

3.2 Creating an Abstract Sentence Recommender

Creating a recommender for selecting sentences for the creation of an extractive summary has four steps. A high-level diagram of the recommender creation process is shown in Figure 3.1.

The first step is to select bug reports that will be used to train the recommendation model. The second step is to extract features from the sentences in the reports. We extracted thirteen features as show in bold in Table 3.1. The third step is to train the text classifier that will classify a sentence as being in a summary or not. We use a logistic regression model to assign a probability value to each sentence. The fourth step is to create the summary. We sort the sentences in descending order according to the probability assigned by the regression model.

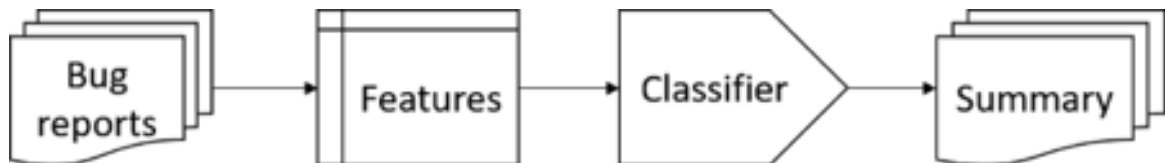


Figure 3.1: Recommender creation process.

3.2.1 Data Source

We use the bug report corpus created by Rastkar et al. [38, 39] in our investigation. The bug report corpus consists of thirty-six (36) bug reports extracted from four open source software projects: Mozilla¹², KDE¹³, Eclipse¹⁴ and Gnome¹⁵. Each bug report in the cor-

¹²bugzilla.mozilla.org, verified 09/12/2018

¹³bugs.kde.org, verified 09/12/2018

¹⁴bugs.eclipse.org/bugs, verified 09/12/2018

¹⁵bugzilla.gnome.org, verified 96/12/2018

Table 3.1: Features used for extractive summarization of bug reports

Category	Feature	Description
Length	SLEN	Word count normalized by longest sentence in the bug report.
	SLEN2	Word count normalized by longest sentence in the turn.
Lexical	MXS	Max S_{prob} score.
	MNS	Mean S_{prob} score.
	SMS	Sum of S_{prob} score.
	MXT	Max T_{prob} score.
	MNT	Mean T_{prob} score.
	SMT	Sum of T_{prob} score.
	COS1	Cosine similarity between sentences using S_{prob} .
	COS2	Cosine similarity between sentences using T_{prob} .
	CENT1	Cosine similarity between sentence and conversation using S_{prob} .
	CENT2	Cosine similarity between sentences and conversation using T_{prob} .
	THISENT	Entropy of the current sentence
	PENT	Entropy after the current sentence
	SENT	Entropy before the current sentence
CWS	Clue word score	
Structural	TLOC	Sentence position in the turn
	CLOC	Sentence position in the bug report
	TPOS1	Time from the beginning of the conversation to the turn
	TPOS2	Time from the turn to the end of the conversation
	SPAU	Time between current and next turn
	PPAU	Time between current and prior turn
Participants	DOM	Participants dominance in words
	BEGAUTH	Is first participant (0/1)

pus has a title which indicates from which of the software projects it comes. Each bug report's comments are given in a format such as that of two or more people taking turns when having a conversation. Therefore, each comment is considered a *turn* and each turn has the participant's name, the time when the conversation started, and the text of the individual sentences from the entire comment. Each bug report has two or more people participating in the conversation. Bug reports are stored in an XML format with each identified by a unique number.

The corpus also contains annotations made by three different annotators for each comment indicating whether the sentence should be included in an extractive summary. Following the procedure given by Rastkar et al. [38, 39], we created a GSS for each bug report by including a sentence in the extractive summary if at least two annotators indicated it should be included.

3.2.2 Extracted Features

Sentence Length Features

The two sentence length features extracted from each bug report comment are SLEN and SLEN2. SLEN is the length of a sentence normalized by the length of the longest sentence in all of the comments in the bug report. SLEN2 is the length of a sentence normalized by the length of the longest sentence in the specific bug report comment.

Probabilistic Weight Related Features

In our investigation, ten probabilistic features were extracted from each sentence of a bug report comment. These lexical features are based on two different conditional probabilities known as S_{prob} (for sentence probability) and T_{prob} (for turn probability).

Equation 3.1 defines S_{prob} . Given a term t by a person making a comment, S_{prob} is calculated by finding the maximum probability of that term's appearance in sentences from all of the comments S . For example, assume there are three commenters for a bug report: A , B and C . If A used the word w_1 seven times, B used w_1 twice and C used w_1 once, then

the maximum probability of w_1 is 0.7, and all instances of w_1 in the bug report receive 0.7 as their S_{prob} weight.

$$S_{prob}(t) = P(S|t) \quad (3.1)$$

Equation 3.2 defines T_{prob} . Given a term t by a person making a comment, T_{prob} is calculated by finding the maximum probability of that term's appearance in a comment T . For example, if a bug report has five comments and the word w_2 appears eight times in one comment, twice in another comment, and in no other comments, then $T_{prob}(w_2)$ is 0.8.

$$T_{prob}(t) = P(T|t) \quad (3.2)$$

Using each conditional probability weight, we calculated three sentence level conditional probability features for the sum of weights, maximum weight and mean of weights. For S_{prob} the features calculated were SMS, MXS and MNS for sum, max and mean respectively. Similarly, for T_{prob} the names were SMT, MXT and MNT.

We calculated four cosine similarity scores for each sentence using S_{prob} and T_{prob} as the word encoding. COS1 and COS2 represented the sentence-wise cosine similarity using S_{prob} and T_{prob} respectively. CENT1 and CENT2 represent the similarity of a sentence to the entire conversation using S_{prob} and T_{prob} respectively.

Clue Word Score

The clue word score was originally developed to summarize e-mails [13]. The authors leverage the quoted text in emails to find dependencies between e-mails. Then they examined the words in the quoted text and original text. If a word, or its root form, was repeated in both the quoted and original text in the emails, such a word is considered as a clue word. A clue word is an indication of the continuation of the same conversation.

Comments in bug reports also have quoted sentences. Developers quote previous comments to request further information or to provide a reply if the quoted text was a question.

```

<Turn>
  <Date>'2008-08-20 10:10:49'</Date>
  <From>'onemen'</From>
  <Text>
    <Sentence ID="13.1"> no problem.</Sentence>
    <Sentence ID="13.2"> current Tabmix dev-build already not disable SessionStore</Sentence>
    <Sentence ID="13.3"> I currently have only one problem , how to disable the restore after restart.</Sentence>
    <Sentence ID="13.4"> can you add a pref for this, or some other way to do it?</Sentence>
  </Text>
</Turn>

<Turn>
  <Date>'2008-08-20 11:04:03'</Date>
  <From>'Simon Bunzli'</From>
  <Text>
    <Sentence ID="14.1"> (In reply to comment #12)</Sentence>
    <Sentence ID="14.2"> &gt; I currently have only one problem , how to disable the restore after restart.</Sentence>
    <Sentence ID="14.3"> You've got several options for that:</Sentence>
    <Sentence ID="14.4"> * Set the prefs browser.sessionstore.resume_from_crash and browser.sessionstore.resume_session_once
    <Sentence ID="14.5"> * Delete the file sessionstore.js as early as possible (e.g. when the profile-after-change notifica
    <Sentence ID="14.6"> * For Firefox 3.1: Respond to the sessionstore-state-read notification by setting the subject's dat
    <Sentence ID="14.7"> Or is there a use case I'm missing?</Sentence>
  </Text>
</Turn>

```

Figure 3.2: Communication thread in bug report (*Firefox:#449596*)

Therefore, Rastkar et al. used the clue word score as one of the features for bug report summarization [38, 39].

Calculating the clue word score is a three-step process. First, the program examined each comment from the beginning of the bug report to the end to find quoted sentences. Quotes were identified using the “>” symbol that appeared in front of a sentence. Once a quote was detected the program traverses backward to find the original sentence which is usually in a previous comment. If the original sentence was found that was extracted. Every non-quoted sentence in the comment where the quoted sentence was found also extracted. This process was carried out until the program reached the end of the bug report.

The second step was to create a graph using the extracted sentences. The nodes in the graph were the sentences. Directed edges were drawn from each non-quoted sentence to the original sentence of the quoted sentence. For example in Figure 3.2 the sentence 14.2 is a quoted sentence and the sentence 13.3 is its original. Therefore, all none quoted sentences in the 14th comment should have draw a directed edge to 13.3 as shown in Figure 3.3. This graph was known as the fragment quotation graph.

Once the fragment quotation graph was created, the third step was to calculate the clue word score for each sentence in the graph. The program traversed through each node and

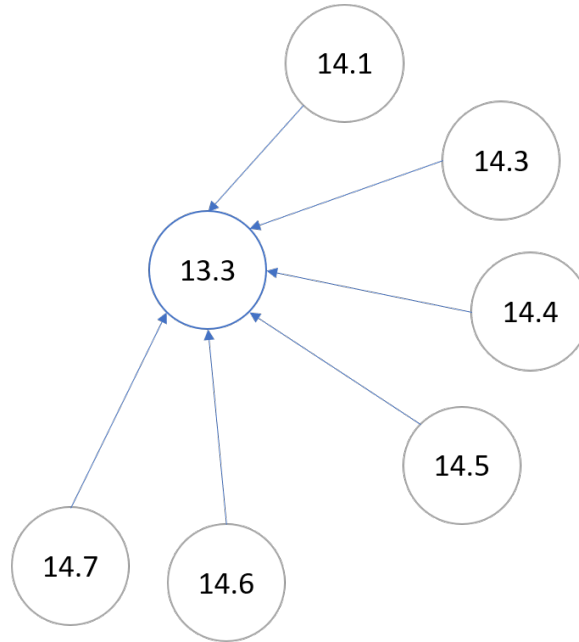


Figure 3.3: Example fragment quotation graph according to the Figure 3.2

examined a node's parent and child nodes to count the number of clue words. Equation 3.3 shows how the clue word score is calculated for word w_i . Equation 3.4 shows how the clue word score is calculated for all clue words in sentence S .

$$CWS(w_i, N) = \sum_{parent(N)} freq(w_i, parent(N)) + \sum_{child(N)} freq(w_i, child(N)) \quad (3.3)$$

$$CWS(S) = \sum_{w_j \in S} CWS(w_j, N) \quad (3.4)$$

3.2.3 Bug Report Summary Creation

A logistic regression model was used to create a classifier which classifies each sentence from the bug report comments as appearing or not appearing in the extractive summary. The Python `sklearn` package was used to implement the sentence classifier. Various logistic regression models were trained to test the effect of the individual feature types and the combined effect of the length and lexical features regarding the performance of the classifier.

Following the procedure outlined by Rastkar et al., an extractive bug report summary was created by selecting the sentences with the highest probability of being included in the extractive summary according to the classifier until the word count of the constructed summary reached 25% of the original bug report’s word count [38, 39].

3.3 Evaluation

To compare our model with Rastkar et al.[38, 39], we trained a logistic regression model using our thirteen selected features. We used the metrics of precision, recall and F-score to evaluate the different models.

Precision measures the percentage of sentences in the extractive summary that were correctly selected compared to the gold-standard summary. It was calculated as the total number of sentences correctly classified as being in the extractive summary divided by the total number of sentences in the extractive summary.

$$Precision = \frac{\# \text{ of sentences correctly selected}}{\text{total \# sentences in the summary}} \quad (3.5)$$

Recall measures how close the generated summary is to the GSS summary. It was calculated as the total number of sentences correctly classified as being in the extractive summary divided by the total number of sentences which appeared in the GSS summary.

$$Recall = \frac{\# \text{ of sentences correctly selected}}{\text{total \# sentences in GSS summary}} \quad (3.6)$$

The F-score value is the harmonic mean of precision and recall.

$$F - \text{score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (3.7)$$

3.3.1 Comparison to the Previous Approach

When we compare the results of our logistic regression model using a 25% word count threshold with that of Rastkar et al.'s [38, 39], we found that our model has an average precision and recall that is less than theirs (Table 3.2). However, this decrease is to be expected, given that thirteen features are used in our model compared to 24 features in Rastkar et al.'s model [38, 39].

Table 3.2: Comparison of our results with Rastkar's results

	Our model	Rastkar's model
Precision	44%	57%
Recall	24%	35%
F-Score	29%	40%

3.3.2 Examining Feature Groups

We examined the model performance when only certain feature groups were used. It was found that when using only the sentence length features (SLEN and SLEN2), the classifier had an average precision of 60%, and the recall was found to be very low at 18%. Using only the probabilistic weights as features (SMS, MXS, MNS, SMT, MXT, MNT), we found that the recall improved to 22%, but the precision declined to 42%. The model which combined both the length and lexical features had a precision of 44% and a recall of 24%. The sentence similarity (COS1, COS2) and the conversation similarity (SENT1, SENT2) were similar to the scores of probabilistic weights. Clue word score (CWS) on the other hand performed well overall considering an F-score of 37%. Figure 3.4 shows the performance results of the different classifiers that we trained.

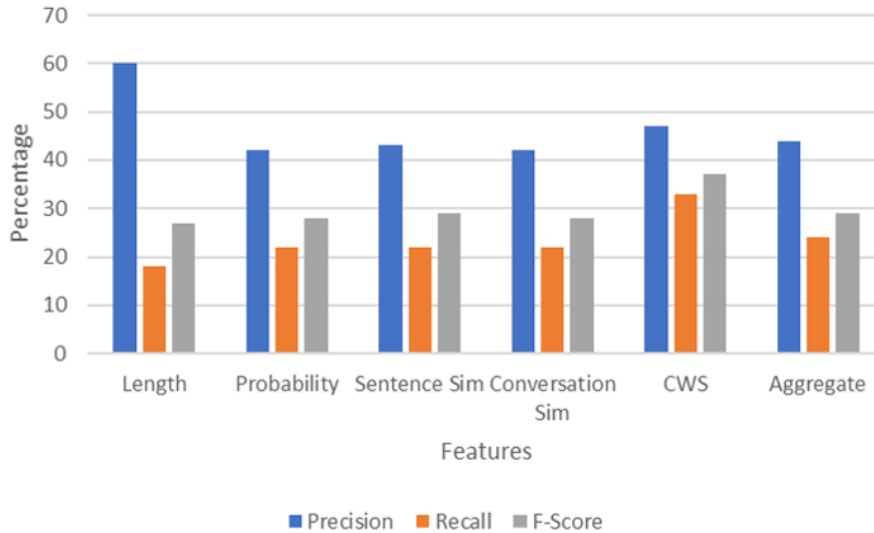


Figure 3.4: Average precision, recall and F-score value of leave-one-out cross validation for classifiers of different feature groups

3.4 Lessons Learned

In training the logistic regression model with the sentence length and lexical features, we found that lengthy sentences often contained useful information needed to create a good summary. This is consistent with the literature which states that the sentence length is used to eliminate short sentences from the summary that do not contain useful information, such as author names or code extractions [27, 28, 42]. However, we also found that only using the length of the sentence was not enough to capture all of the useful sentences, as selecting longer sentences resulted in quickly reaching the word percentage threshold. Our results for the use of lexical features also show that the recall is higher when the length is not considered, as the classifier selects more of the shorter sentences thereby increasing the recall.

As we are taking a supervised learning approach to summary creation, the results are sensitive to the summaries found in the GSS. When comparing the word count of the summaries in the GSS for each bug report, we found that some summaries had a word count of more than 50% of the original bug report’s word count. As we used a word count threshold of 25% of the original bug report’s word count for the summaries, our model is unlikely to

choose all the sentences found in the GSS summary for some of these summaries and this results in a low recall for the trained models.

Rastkar et al. hired three annotators to create extractive summaries. They used the kappa test to measure the agreement level between annotators. The annotator agreement score was 0.41 for their study [38, 39]. The disagreement levels indicated the different levels of information requirements. If the human annotators cannot agree among themselves above 50%, achieving an automatic summarization solution that has high precision or recall is not a realistic goal.

Our replication study confirms that certain features were very important in selecting sentences to a summary. As we implemented only half of the features and we did not implement sentence entropy related features those with higher variability score [38, 39], our evaluation scores were lower. Even if we implemented the rest of the features, it only confirms that we could train a moderately performing summarizer that does not work very well for bug reports outside the training corpus. Rastkar et al. also confirmed that their model was sensitive to the data in the training corpus.

3.5 Summary

We partially replicate the work of Rastkar et al. [38, 39] to evaluate the contribution of different features to create a bug report summary. We learned that certain features were useful in selecting summary sentences compared to others. We found that long GSS and moderate agreement level between annotators affect the performances of the classifier. Our findings of the feature evaluation study direct us to take a different approach to create bug report summaries.

Chapter 4

Automatic Sentence Annotation

In chapter 2 we provided an overview of bug report summarization studies. Even though unsupervised bug report summarization performed better than supervised models, fixed-size summaries missed information required by certain software developers. In this chapter, we provide a labelling technique that identifies the content in bug report comments. Then later in the chapter, we introduce an interface that allows users to customize the content that they want to see based on the labels.

4.1 Schemas for Bug Report Summarization

Bug reports contain structured and unstructured data [7]. Most of the comments appear as free text, a form of unstructured data. Structured data in bug reports include code snippets, stack traces, error reports, and attachments [8]. After careful examination of the bug reports in Rastkar et al.'s corpus [38, 39], we developed a bug report schema for summarization, as shown in Figure 4.1. According to this schema, a bug report is composed of sentences. The content of these sentences could be of interest to software engineers and we can use features and keywords to identify the interesting sentences. Some of the sentences have dependencies on other sentences in the bug report. We can use these dependencies between sentences to apply techniques such as clue words [13] and topic modelling.

The sentences in bug report comments are created with various intentions. We developed a schema for categorizing the four common intentions that we observed in bug reports, as shown in Figure 4.2.

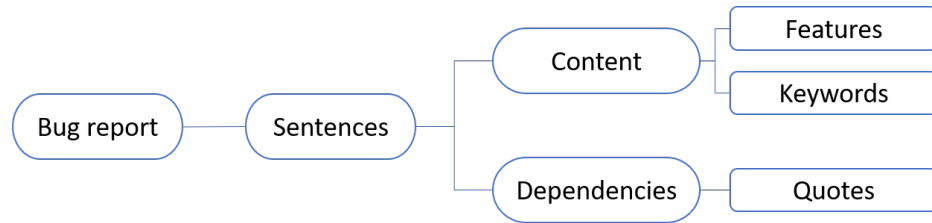


Figure 4.1: Bug report summarization schema

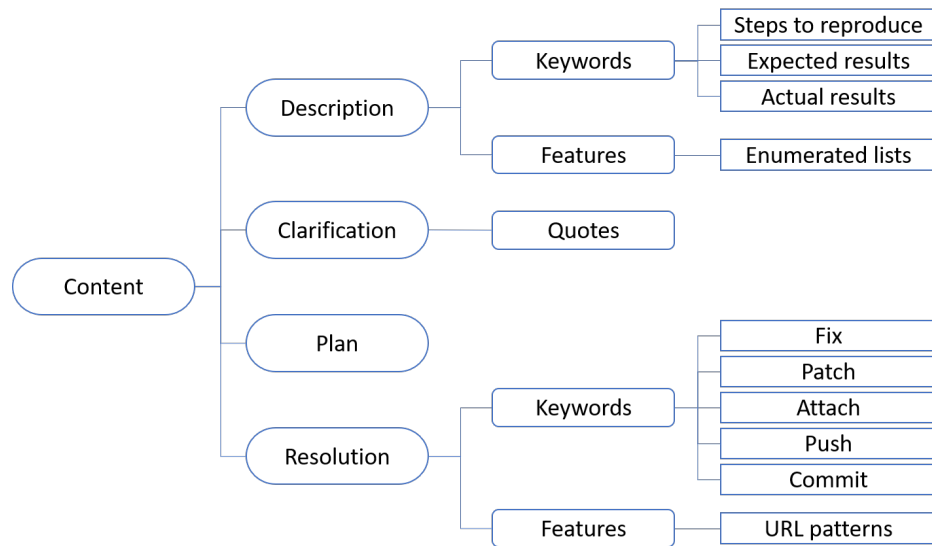


Figure 4.2: Bug report sentence intention schema

The first comment in a bug report is always the description of the bug. The description provides brief details about the bug. Some bug descriptions are written as a paragraph (i.e. unstructured) and others are well-structured. In all cases, the three intentions of *steps to reproduce*, *actual outcome*, and *expected outcome* are found. In a well-structured bug report description, these intentions were easy to identify. We found that recently reported bug reports, after year 2004 in Rastkar et al.'s [38, 39] corpus, were more likely to contain this structure. However, when examining the bug reports in the their corpus, where some bug reports were reported as early as the year 2000, automatically identifying these intentions was more challenging as the descriptions were mostly of the paragraph form.

Other developers post comments once a bug is reported. Upon careful examination of the comments subsequent to the first comment (i.e. bug description), we found three intentions.

First, if a bug description or any other comment is not clear enough, developers request further information with a response appearing in a later comment. We identify such comments as having an intention of *clarification*. As developers typically quote the original sentence(s) from the previous comment(s) when they request further information or when they respond to a request, this creates a communication structure similar to that of email threads. Therefore, we can apply the “clue word score” [13] algorithm to identify the connections between these sentences.

Second, sometimes when developers identify the problem, they create a solution. We refer to this intention as *resolution*. We use two techniques to identify comments with this intention: keywords and hyperlinks. We found that *resolution* comments often contain keywords such as “fix”, “patch”, and “attachment”. In some cases, the comments contain links to specific commits in the version control system. We used uniform resource locations (URL) patterns to recognize such comments as also being *resolution* comments.

Lastly, a developer may introduce an idea to solve an issue but not provide the actual solution as with a *resolution* comment. In this case, the comment is considered to have a *plan*

intention. We do not yet have a specific pattern or set of keywords to identify such comments due to the diversity of scale for the problems described in bug reports and developers' personal preference when composing comments. For example, if the proposed solution to a problem is small, such as changing a parameter or adding one line to the source code, the comment could be a single sentence. On the other hand, a proposed solution may be complex and have multiple lines of code snippets and/or comments. Also, developers express their *plan* in a variety of ways, such as enumerated lists and paragraphs. Consequently, we were unable to derive a general enough heuristic to assign the *plan* label to comments with a sufficient level of accuracy.

4.2 Automatic Labelling of Bug Report Comments

Now we present our approach for detecting and labelling the content in bug report comments. First, we describe the dataset used to create and test our approach. Next, we present the content labels and how these labels are applied.

4.2.1 Data Set

To develop and test our approach to automatically labelling bug report comments, we chose to use the bug report corpus curated by Rastkar et al. [38, 39]. This choice was made because the dataset is publicly available¹⁶ and is the bug report corpus used in previous two studies by Mani et al. [31] and Lotufo et al. [30].

The bug report corpus is a collection of XML (Extensible Markup Language) documents with the following schema:

¹⁶<https://github.com/HuaiBeibei/IBRS-Corpus> verified 01/05/2020

<BugReport> The contents of a bug report.

<Title> The bug report title, product name, and the bug report ID, as given by the bug tracking system.

<Turn> A turn represents a comment by a developer. A <Turn> element contains three elements:

<Date> The date of the comment.

<From> The name of the developer who wrote the comment.

<Text> The text of comment. The <Text> element contains one or more <Sentence> elements.

<Sentence> A line in the comment. This could be a proper sentence, a line of code or an empty line. All <Sentence> elements contain an ID attribute.

4.2.2 Data Extraction and Processing

We extracted each bug report from the corpus and processed the data in the following manner:

1. The contents of <Sentence> elements were preprocessed using various techniques. The `text-clean` module that we implemented provides the following functionalities. Each function takes a parameter to allow for customizing the cleaning process as needed. For example, when detecting URL, symbols such as `.` and `\` are required. Therefore, one can set the parameter `remove_punc = False` to keep these symbols.

Remove punctuation We use the list of symbols in Python's `string.punctuation` to match and remove punctuation symbols in the text.

Remove digit Python's `string.digits` provides the list of digits 0 to 9. We apply a string translate function to remove digits from the text.

Remove emojis Although comments in bug reports seldom contain emojis, we do remove them in cases where they occur. To remove emojis we use the `emoji`¹⁷

¹⁷<https://pypi.org/project/emoji/> verified 11/06/2020

package developed for Python.

Convert HTML escape characters The corpus contains double escaped HTML characters. For example, `>` is written as `>`. Therefore, we run an `unescape` function twice to convert such text. We use the Python `html`¹⁸ package to convert escaped characters.

Convert escape characters As the corpus contains escape characters, such as `\`, we use Python package `codec`¹⁹ to decode these characters.

Remove stop words To identify stop words, we use the smart stopword list created by Nayeem et al. [34], as it has more words than the regular stop word list provided by the `nltk` English word list. All words in the corpus are converted to lower case in this function because all words in the stopword list are in lower case.

Lemmatize words Word lemmatization is used to get the root form of a word. For example, *carry* is the result of lemmatizing the word *carries*. We used the `WordNetLemmatizer`²⁰ from Python's `nltk` package. We prefer lemmatizing over stemming because stemming does not provide the actual root word. Using the same example as above, the output of stemming would be *carri*.

2. The original text is passed to the following modules for labelling. The order of calling the `URL Detector`, `Code Detector`, and `Off-Topic Classifier` is important because the `Code Detector` and `Off-Topic Classifier` depend on the outcome of the previously called modules. Note that the rest of the modules are independent of each other, and the order of use is not important.

URL Detector All comments are passed into this module to find sentences containing URLs. Since most of the URLs point to files or commits in a version control

¹⁸<https://docs.python.org/3/library/html.html> verified 11/06/2020

¹⁹<https://docs.python.org/3/library/codecs.html> verified 11/06/2020

²⁰https://www.nltk.org/_modules/nltk/stem/wordnet.html verified 11/06/2020

system (e.g. Subversion) or other reports in the bug tracking system, a developer can use these links to examine these file or bug reports as needed.

Code Detector The code detector identifies syntax, error messages, stack traces, and preference parameters in all of the comments. This module ignores the sentences labelled by the URL Detector.

Off-topic Classifier A Naïve Bays classifier was trained to detect off-topic sentences using comments extracted from YouTube and StackOverflow. Comments from YouTube were used as positive examples (i.e. off-topic sentence) whereas comments from StackOverflow were used as negative examples (ie. on-topic sentences). Digits and punctuation symbols are removed from each sentence before they are passed to this classifier. This module ignores sentences labelled by the URL Detector and Code Detector.

Clue-word Detector The entire bug report is passed in to determine the clue-word relationships between sentences for detecting the *clarification* intention.

Topic Modeller Topic words are extracted from each bug report and each sentence is compared against the created topic word list. Since topic words represent the dominant themes in a discussion, sentences containing these words have a higher significance over other sentences. This information is useful as every developer does not quote comments when responding to a previous comment.

Reproduction Step Detector Only the first comment (i.e. bug report description) is passed into this module. The module finds the sentences that describe the bug and its reproduction steps for detecting the *steps to reproduce* intention.

Resolution Detector All comments, except the first one, are passed into this module to identify sentences with the *resolution* intention.

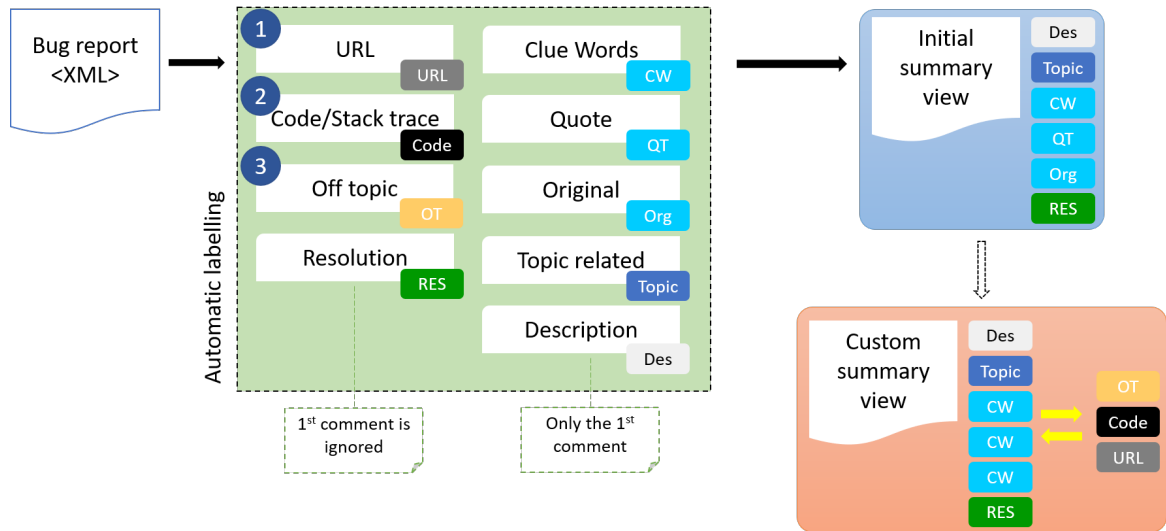


Figure 4.3: Overview of the labeling and summarization process

Table 4.1: Description of labels

Label	Description
Des	Bug report description including steps to reproduce, expected outcome and actual outcome
CW	Clue words
Org	Original sentence if the sentences is quoted in later comment
QT	Quoted sentence from a previous comment
Topic	Topic word is included
Res	Resolution statements (contain fix, patch, attachment)
OT	Off-topic sentence
URL	Sentence contains a URL
Code	Codes, stack traces, error messages and etc.

4.2.3 Automatic Assignment of Labels

We derive nine labels to identify the content found in bug reports. The labels are shown in Table 4.1. The first six labels are assigned to sentences that are considered to contain useful information for bug report users. The last three labels are assigned to sentences considered to be unimportant or noise, which can then be filtered out as desired.

Description [Des] The bug report's description contains essential information for software developers. For example, steps to reproduce, observed results, and expected results

are all useful to triagers when determining duplicate bug reports or to developers when creating a fix. We apply the label `Des` to a sentence in the first comment if it is determined to contain steps to reproduce the bug, observed results or expected results. Regular expressions are used to capture the explicitly stated steps to reproduce, observed results, and expected results. If these regular expressions fail, a regular expression for identifying any enumerated lists is used, with the assumption that such a list is describing the steps for reproduction. Finally, the sentences in the bug report description are compared with the bug report title. If there is a common word in both the sentence and the title, we apply the `Des` label to the sentence. Note that we remove stopwords and lemmatize the words to identify variations of the same word that appears in the comments and the title.

Clue Word [CW], Original [Org], and Quoted [QT] Developers often quote comments from other developers when they need further information about that comment or when they need to respond to that comment. For example, if a developer posts a question, another developer will post the answer by first quoting the question and then replying. Or if a developer posts a way to solve a bug, another developer will post a comment accepting the solution or requesting more information by quoting the solution. As these quotes and responses create a chain-like communication between developers, we use the labels `Org` for an original sentence (i.e. start node) and `QT` for when that sentence is quoted (i.e. end node). The label `CW` is attached to a sentence if both the response and the quote have words in common. `CW` denotes the presence of “clue words” as proposed by Carenini et al. [13] for email summarization.

Figure 4.4 shows a portion of a bug report from Rastkar et al.’s corpus [38, 39] that contains a communication thread. In *Turn 5*, there are two quoted sentences (sentences 5.2 and 5.3) from *Turn 4*. The two sentences in *Turn 4* receive the `Org` label and the sentences 5.2 and 5.3 in *Turn 5* receive the `QT` label. The unquoted sentences in *Turn 5* are possible responses. If a sentence has a common word (or at least a common root form) with a quoted sentence, the unquoted sentence receive a `CW` label. Here, the sentence 5.5 has the

```

</Turn>
<Turn>
  <Date>'2009-06-19 12:07:34'</Date>
  <From>'Gavin Sharp'</From>
  <Text>
    <Sentence ID = "4.1"> (From update of attachment 383211 [details])</Sentence>
    <Sentence ID = "4.2"> Perhaps we should rename one of them to _fhResult just to reduce confusion?</Sentence>
  </Text>
</Turn>
<Turn>
  <Date>'2009-06-19 13:17:56'</Date>
  <From>'Justin Dolske'</From>
  <Text>
    <Sentence ID = "5.1"> (In reply to comment #3)</Sentence>
    <Sentence ID = "5.2"> &gt; (From update of attachment 383211 [details] [details])</Sentence>
    <Sentence ID = "5.3"> &gt; Perhaps we should rename one of them to _fhResult just to reduce confusion?</Sentence>
    <Sentence ID = "5.4"> Good point.</Sentence>
    <Sentence ID = "5.5"> I renamed the one in the wrapper to _formHistResult. </Sentence>
    <Sentence ID = "5.6"> fhResult seemed maybe a bit too short.</Sentence>
  </Text>
</Turn>
<Turn>
  <Date>'2009-06-19 13:20:52'</Date>
  <From>'Justin Dolske'</From>
  <Text>
    <Sentence ID = "6.1"> Pushed http://hg.mozilla.org/mozilla-central/rev/097598383614</Sentence>
  </Text>
</Turn>
</BugReport>

```

Figure 4.4: Communication thread in bug report (*Firefox:#495584*)

word renamed and the quoted sentence 5.3 has the word rename. Therefore, the sentence 5.5 receives the label CW.

How a clue word score is calculated for bug reports was explained in Section 3.2.2. As we are interested in the sentences that have clue words, not the number of clue words, we add the label CW to a sentence that has at least one clue word.

[Topic] In Rastkar et al.’s corpus [38, 39], we observed that developers sometimes respond to other comments without quoting the previous comment(s). As the communication threads are not explicit, we can not directly identify such communication patterns in the bug report, resulting in our automatic labelling approach missing important information and the comments not receiving a label.

To address this problem, we chose to identify comments that participate in a common topic, specifically the dominant topic in the bug report. To do this, we applied a topic modelling approach from natural language processing. Topic modelling is a technique used to identify topics from a pool of documents from different fields [9]. However, we are not interested in identifying the different topics in a bug report in the work. Instead, we utilize

the technique to capture the dominant topic words in the bug report by extracting only one topic from the topic model for each bug report. Then we compare the comments against the set of topic words to find comments that participated in the dominant conversation of the bug report. We assign the label `Topic` to these sentences.

We use the Latent Dirichlet Allocation (LDA)²¹ model in the Python `gensim` package to extract topic words from a bug report. We remove sentences labelled with `OT`, `Code`, or `URL` as those do not contain useful information for topic extraction. The remaining sentences are passed through our text cleaning module before creating the bag of words for the bug report.

Resolution [Res] Once a bug report is created, it goes through different states, as indicated by the report’s status, with the actual states differing between bug tracking systems [3]. During these different states, developers try to reproduce the bug, provide screenshots or mock-ups of user interfaces, create partial solutions, or create patches that may be accepted as the solution by another developer. Such attachments, partial patches and final solutions are of interest to bug report users [7].

We use regular expressions to identify attachments and comments that reflect the final solution or an intermediate step towards the final solution. The label `Res` is assigned to a sentence if that sentence has such keywords as `push`, `fix`, `patch`, `attach`, `attachment`, and `commit`. Similar to clue word matching, we do root word matching instead of exact word matching.

When examining Rastkar et al.’s corpus [38, 39], we found comments such as “Patch 3” and “Previous patch with some minor adjustments.” (from bug report *Eclipse* #224588). Therefore, we use the regular expressions `^patch.*` and `.*patch.*` to capture such comments. For the other regular expressions, we replace `patch` with the other keywords previously mentioned to identify comments that contain those keywords.

²¹<https://radimrehurek.com/gensim/models/ldamodel.html> verified 12/05/2020

Off-Topic [OT] During our inspection of the bug reports in Rastkar et al.’s corpus [38, 39], we found comments which did not contribute to the ongoing conversation. Those comments were usually greetings or appreciations towards another developer, such as “*Hi John*”, “*Thank you*”, “*Great!*”, or “*Nice work!*”. The label OT is assigned to such a sentence to reflect that it is an off-topic comment in the bug report.

To determine the off-topic comments, we followed Chowdhury and Hindle’s approach for filtering off-topic comments from Python IRC channels [16]. We extracted comments from YouTube²² and StackOverflow²³ (SO) as positive (i.e. off-topic) and negative (i.e. on-topic) examples, respectively, to train a classification model. The YouTube dataset consisted of comments from YouTube channels with the keywords `English teaching`, `cooking`, `news` and `sports`, as these were considered to likely contain a good representation of common off-topic comments with respect to software development. StackOverflow was chosen because it has millions of questions and answers, where both the questions and answers have comments to represent software development conversations. As posts in SO are regulated by the community, we expected there to be fewer off-topic comments in SO comments. We extracted ~ 3000 lines of comments from each of YouTube and SO as positive and negative examples, respectively. We manually filtered the comments from SO as we did find a few greeting comments in that data. We excluded those comments when training the models. During preprocessing we converted all text to lower case and removed all punctuations symbols, extra white spaces, and stopwords. Then we lemmatized the words to remove the effect of variations of the same word. We trained a Support Vector Machine (SVM) classifier and a Naïve Bayes classifier. We used the SVM model to categorize the bug report comments into off-topic or on-topic comments because we found that the Naïve Bayes classifier produced too many false positives. The results for both of these classifiers are presented in Section 5.1.2. Note that this is the only sub-module that has a supervised learning model. However, we made sure the classifier is not sensitive to the training dataset

²²<https://developers.google.com/youtube/v3/getting-started>

²³<https://api.stackexchange.com/docs>

by not using bug report comments. We believe our decision to follow Chowdhury and Hindle’s [16] approach improves the generalizability of our model.

[URL] Uniform Resource Locators (URLs) are often added to bug reports by either developers or DevOps tools. For example, we observed links to other bug reports and to version control commits. Links to other bug reports allow developers to investigate the content of these other bug reports. This may help a developer in solving a current bug, or a triager to create a “super bug” that gathers together a set of reports that have similar behaviour. Links to version control systems, such as Git and Subversion, allow developers to inspect the changes made by a patch. Although URLs may be considered as noise for summarization, a bug report user could benefit from having these links accessible when a deep investigation of all of the information is needed. Therefore, we consider being able to identify URLs automatically and allowing users to view them as important.

We use regular expressions to identify URLs in comments and assign them the label URL. During our investigation of Rastkar et al.’s corpus [38, 39], we found URLs for bug reports in the bug tracking system, commits to the version control system, and web pages to access information or download software. In general, all URLs used either `http(s)` or `ftp` as the protocol. Therefore, we wrote two simple regular expressions, `.*https?://.+` and `.*ftp://.+`, to identify URLs.

[Code] Bug report comments can contain code snippets, error messages, software configuration parameters, stack traces and software version names and numbers. Although this information may be relevant to a bug report, it may also not be essential for understanding the bug report. Mani et al. [31] identified such information as noise in their study because the summarization models performed poorly when this noise was present. We assign the label `Code` to such sentences so that the user can decide whether to show or hide such information in the summary view.

We investigated two approaches to identify comments with such technical information.

```

(ROOT
  (S
    (VP (VB Feel)
      (ADJP (JJ free)
        (S
          (VP (TO to)
            (VP (VB open)
              (NP (DT a) (JJ new) (NN bug))
              (SBAR (IN if)
                (S
                  (NP (EX there))
                  (VP (VBZ is)
                    (NP
                      (NP (NN something))
                      (SBAR
                        (S
                          (NP (PRP I))
                          (VP (VBD missed))))))))))))))
          (. .)))
  )
)

```

Figure 4.5: Syntax parse tree for sentence “Feel free to open a new bug if there is something I missed.”.

Our first approach was to use the syntax parse tree structure of a sentence. For a grammatically correct sentence in the English language, the parse tree has the root node *S* to represent a sentence, as shown in Figure 4.5²⁴. As code snippets, stack traces, and error messages don’t follow the rules of English grammar, their tree structure does not have a node *S* (Figure 4.6) and this information can be used to identify them. However, we found that this method also identifies short sentences as *Code* because such sentences do not follow grammar rules. For example, “Actual results” and “Fair point” were labelled as *Code* by the module. Even though those sentences were perfectly meaningful to humans, for the computer model, they were not. This led us to look into an alternative solution.

²⁴The tree was generated using <https://stanfordnlp.github.io/CoreNLP/corenlp-server.html> verified 14/06/2020


```

(ROOT
  (NP
    (NP (JJ public) (JJ static) (NN void))
    (NP
      (NP (JJ main))
      (PRN (-LRB- -LRB-))
      (NP (NNP String) (NN ar) (-LRB- -LSB-) (-RRB- -RSB-))
      (-RRB- -RRB-))))))

```

Figure 4.6: Syntax parse tree for code `public static void main(String ar[])`.

Our second approach was to use a set of regular expressions to capture code snippets, stack traces, error messages, and other software related configuration parameters. As our dataset contained bug reports from four software projects, the programming languages used, the error messages returned, the structure of stack traces, and the configuration settings parameters of the tools were all different. Therefore, we wrote various regular expressions to capture different types of patterns. We found that writing multiple simple regular expressions such as `\S+.[ch].*` was sufficient when compared to complex regular expression such as `(\w+\/?)+[a-z\-\]+\.\.(h|cpp|c|java|py|html|css|js).*`.

All of the regular expressions were written to a text file. This was done to all for modification of the existing regular expressions or to add new patterns without having to change the software. We passed the original sentences from the bug report comments through the code identifier without any preprocessing. Then each sentence (or line) was matched against the patterns in the file to detect codes in comments.

4.3 A Task Relevant Bug Report Summarizer

Having labelled the bug report sentences by their content type or intention, an interface can be provided to the bug report users which presents the labelling and allows the user to filter the sentences according to their information needs. Figure 4.7 provides an example of the user interface for such a tool that supports the creation of task-relevant bug report

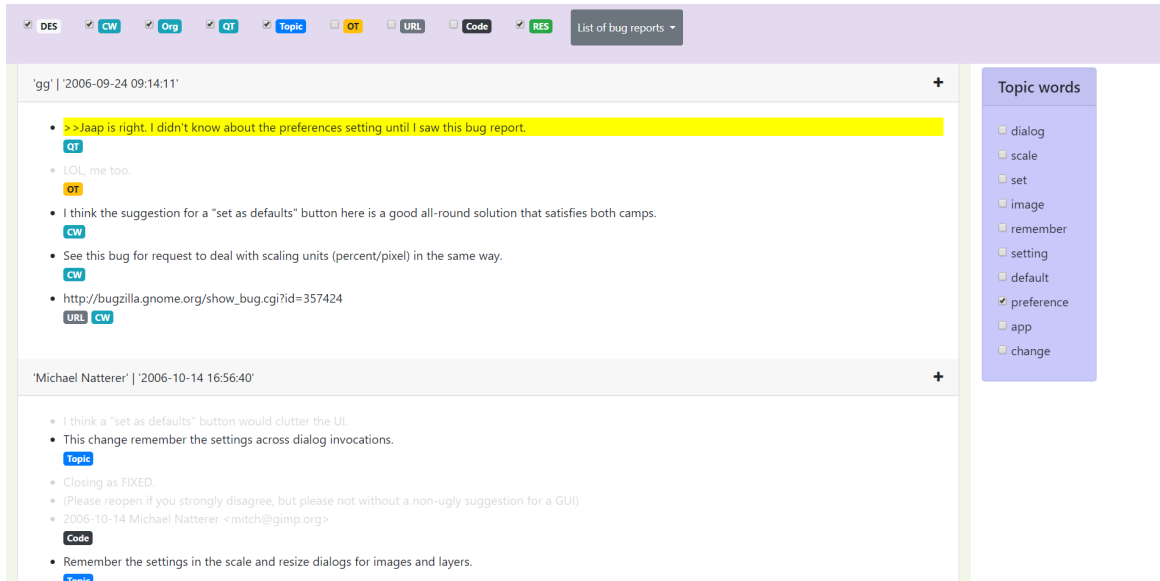


Figure 4.7: User interface of customizable bug report summary

summaries. The interface is intentionally modelled after Mozilla’s Bugzilla²⁵ interface to leverage user familiarity with that or similar systems.

The tool provides access to all of the comments from the bug report instead of displaying only the automatically selected sentences as in the traditional bug report summarizer. Similar to Bugzilla and other issue tracking systems, the user can choose to expand or collapse individual comments at will.

At the top of the interface are checkboxes for the various labels, allowing the user to show or hide sentences according to the particular sentence label. On the right-hand side is provided a list of the dominant topic words within the bug report, allowing the user to filter the sentences by topic.

As mentioned, sentences are displayed with a label indicating to which sentence category it belongs. The labels are colour-coded to aid in understanding the relevance of the sentence category. For example, the *Resolution* label is green to indicate a high perceived relevance to a user, and the *Off-topic* annotation is yellow to indicate that the sentence is likely not relevant to the user. In Figure 4.7, the user has selected to focus on sentences

²⁵<https://bugzilla.mozilla.org/> verified 20/05/2020

with the labels *Des*, *CW*, *Qt*, *Org*, *Topic*, and *Res*. Sentences that are visible, but are not one of these categories, are shown greyed out, such as “*LOL, me too*” which has an *OT* label.

When the user selects a topic word that they are interested in, the interface highlights the word in the comments to focus the user’s attention. In Figure 4.7 the user-selected preference in the topic word list, and the comments with this word are highlighted (e.g. the sentence at the top of the screen).

4.4 Summary

In this chapter, we introduced a bug report summarization schema and a bug report sentence intention schema. The schemas were used to identify the content in sentences in bug report comments. We used regular expressions and substring matching to automatically identify the content. Our labeling submodules were applied to bug reports extracted from four bug tracking systems. The *off-topic* classifier is the only module that has a supervised learning model. Since we did not use bug report data to train the model, we maintained its generalizability. Finally, we introduced a task-relevant bug report summarizer that allowed developers to create a custom bug report summary according to their information needs.

Chapter 5

Results and Discussion

In this chapter, we presented our results and discussed our findings of each sub-module. We have given examples from the bug report corpus to better understand why some sentences were incorrectly labelled.

5.1 Results

To develop and test our approach for automatically labelling bug report sentences, we chose to use the bug report corpus curated by Rastkar et al. [38, 39]. This bug report corpus is a collection of XML documents where sentences are identified by a `<Sentence>` tag. Figure 5.1 shows a portion of the *Firefox:#449596* bug report from this corpus.

The choice to use Rastkar et al.'s [38, 39] bug report corpus was made for two reasons. First, the dataset is publicly available and is the bug report corpus used in the previous two studies of bug report summarization by Mani et al. [31] and Lotufo et al. [30]. Second, as the comment sentences are already identified and verified, this removes errors that may occur by a sub-optimal sentence tokenisation approach on a raw bug report dataset.

5.1.1 Manual Labelling of Sentences

To evaluate the quality of our automatic sentence labelling approach, four volunteers were asked to annotate by hand the sentences in Rastkar et al.'s corpus with the labels from Section 4.2.3. One annotator examined and labelled the entire corpus. The other volunteers were assigned non-overlapping thirds of the corpus to label independently. Once the labelling was completed, the first annotator met individually with the other annotators

```

<BugReport ID="2">
  <Title>
    "(449596) Firefox - remove the
    browser.sessionstore.enabled pref "
  </Title>
  <Turn>
    <Date>'2008-08-07 07:22:12'</Date>
    <From>'Simon Bunzli'</From>
    <Text>
      <Sentence ID="1.1">
        That pref was thought to be for
        extensions which wanted to completely replace
        our own Session Restore functionality.
      </Sentence>
      <Sentence ID="1.2">
        While this has worked somehow for Tab
        Mix Plus, we've .....
      </Sentence>
      .....
    </Text>
  </Turn>
</BugReport>

```

Figure 5.1: A portion of *Firefox:#449596* bug report in Rastkar et al.'s corpus.

to resolve any disagreements. The agreed upon annotations were then compared with the labels assigned by the approach.

5.1.2 Results of Automatic Labelling

Table 5.1 shows the precision, recall and F-score for each type of label assigned by our approach. The remainder of this section provides a discussion for each labelling module of these results. Equations 5.1 and 5.2 show how the precision and recall were calculated for each label.

$$Precision = \frac{\text{\# of labels correctly assigned}}{\text{total \# labels assigned}} \quad (5.1)$$

Table 5.1: Quality of automatic annotations

Label	Precision	Recall	F-Score
Des	90.91%	48.34%	63.12%
CW	65.33%	63.64%	64.47%
Org	92.31%	98.63%	95.36%
QT	92.68%	98.7%	95.6%
Res	83.97%	77.06%	80.37%
OT (SVM)	58.23%	50%	53.8%
OT (Naïve Bayes)	54.43%	50.59%	52.44%
URL	86.84%	80.48%	83.54%
Code	63.81%	45.12%	52.86%

$$Recall = \frac{\# \text{ of labels correctly assigned}}{\text{total \# labels assigned by annotators}} \quad (5.2)$$

5.2 Discussion

5.2.1 Description Labeller

The `Description Labeller` achieved a 91% precision with a recall of 48%. After analyzing the false negatives, we found that the automatic annotator missed the root form similarity of words such as *rescale* and *scaling*. For example, in bug report *GIMP:#164995* the term *scale* was in the title. The words *rescaled* and *scaling* appeared in the bug report description. But the module lemmatizer failed to find the root form of those words as *scale*. Therefore, the label *Des* was not assigned.

Most of the bug reporters used plain text to describe expected and actual software behaviour in the description. However, some reporters included stack traces along with other text. This resulted in the `Code Detector` identifying code fragments and other program-related sentences that were part of the description. However, recall that the `Description Labeller` uses three different strategies to assign a *Des* label: words in common with the title, enumerated lists, and structured text. As a result, the `Description Labeller` would assign the *Des* label to the same sentences as those identified by the `Code Detector`, if

the sentences appeared after the identified *steps to reproduce*, *actual behaviour*, and *expected behaviour*. For example, lines 18-151 of the description in *Bugzilla:#429126* is a stack trace. As the *expected behaviour* is explained in lines 15-17, the `Description Labeller` labelled every sentence afterwards (i.e. the stack trace) as *Des*. In other words, the `Description Labeller` falsely labelled 133 sentences in this bug report.

When investigating the reasons for a low recall rate (48.34%), we found that the `Description Labeller` failed to distinguish between the use of some words from code, whereas a person would make this distinction. For example, the title of *Eclipse:#260502* has the word `class` inside brackets (i.e. "class") and the word `class` is repeated in the first comment. To a person, it is obvious that the reporter is describing class documentation, however to the `Description Labeller` these sentences appeared to be a piece of code and was not labelled.

5.2.2 Clue word Detector

The `Clue word Detector` achieved a 65% precision and a 64% recall. We found that the detector did well at capturing original (*Org*) and quoted (*QT*) sentences. However, capturing clue words (*CW*) was not as effective.

When investigating the clue words missed by the detector, we found that certain words were not converted to the expected root form by the lemmatizer. For example, *localization* was not converted to the same root as *localize*. Although a person can identify such words as clue words, the `Clue word Detector` failed to identify such words that had similar semantics.

5.2.3 Off-topic Classifier

The `Off-topic Classifier` trained using the SVM algorithm had a precision and recall of 58% and 50%, respectively, whereas the `Off-topic Classifier` training using a Naïve Bayes algorithm was a bit worse with a precision and recall of 54% and 51%, respectively.

When training the off-topic classifier, we were careful about choosing our training data. The positive data set extracted from YouTube contained greeting sentences, which were usually very short (e.g. “Thank you”, “Thanks”, and “Much appreciated”). With the bug report comments, we found that the classifier successfully detected short greetings similar to the greetings in the training data, but the classifier failed when the sentences were long. For example, in *Eclipse:#224588* the sentence “Again, I think it’s a brilliant idea Martin.” was not correctly labelled as being off-topic.

5.2.4 Code Detector

Our `Code Detector` achieved precision and recall values of 64% and 45%, respectively. Regarding these values, when detecting code, the regular expression patterns in the `Code Detector` captured syntax embedded in regular English sentences as opposed to proper code snippets. Such sentences were not marked as code by the human annotators because they observed that the programmers were discussing the code, not providing code for context or as a solution.

When creating our labelling modules, we chose not to remove symbols and stop words which are important for detecting programming language syntax. This resulted in some sentences that contain brackets, parenthesis, and special symbols, which are common in programming language syntax, being detected as code. For example, *Eclipse:#223734* contains the sentence “According to the JavaDoc in `Platform.getResourceString(Bundle,String)...`”, which was labelled as code according to the regular expression `(\S+\.?)+\ (\S+, \S+)\ .*` used in the module. Another example is *Eclipse:#260502* which describes an issue in a JavaDoc file. The reporter provided an example of the order of defining a set of classes and interfaces. As these comments contain the words *class* and *interface* we could not write a regular expression pattern that identifies such a comment as code, but also correctly excludes sentences that contain the same words. Such situations resulted in the 45.12% recall rate for the `Code Detector`.

5.2.5 Resolution Detector

We found our Resolution Detector to be effective with precision and recall rates of 84% and 77%, respectively. Recall that the Resolution Detector searched for keywords such as *patch*, *push*, *commit*, *attach*, and *fix*. Most of the comments that contained such keywords, in their various forms, were found by the human annotators to describe either the solution or an intermediate form of the final resolution for the bug report. However, certain sentences were questions that inquired about a patch or a fix, or a further explanation that contained one of the keywords. For example, *Eclipse:#154119*, contains the sentence “*It would be possible push parts of TextFieldNavigationHandler down to Platform Text...*”, which contains the keyword *push* but is not a resolution statement.

5.2.6 URL Detector

The URL Detector was effective in correctly identifying links, with a precision of 87% and a recall of 80%. Although URL regular expressions are well-known and developed, the reason that this detector did not achieve as a higher score was due to URL patterns in stack traces. We found that some stack traces contained the text `http://localhost` which were labelled as *URL* by the module. Such URLs were not considered useful to bug report users by the annotators.

5.3 Summary

We evaluated the automatically assigned labels with manually annotated labels in the modified bug report corpus. Some of the models received good results while some of those received moderate results. Except for the *off-topic* classifier, all of the annotating modules that we developed are unsupervised models. While our annotation models have room to improve we believe this is a good starting point to create customized bug report summaries.

Chapter 6

Conclusion and Future Work

In this study, we focused on creating customizable bug report summaries. Our rationale for having customizable summaries is that fixed-length summaries cannot support the different information needs of diverse software development roles. We analyzed the contribution of different features to create a bug report summary by partially reproducing Rastkar et al.'s supervised learning model [38, 39]. Our findings and the comments of the user studies conducted by Rastkar et al. [39] and Lotufo et al. [30] made us take a different approach to bug report summarization. Since different users are interested in a variety of information, we developed two schemas to identify the intentions of sentences in bug report comments. We developed a set of labelling modules to identify the content or intention of sentences in bug report comments according to the schemas that we developed. The empirical results indicate that in general, these labelling modules achieve sufficient precision and recall scores for practical use. The code detector and off-topic classifier received slightly over 50% F-score. Even though those scores were not as high as the other sub-modules, they provide a good first step in the direction of bug report sentence labelling.

6.1 Future Work

In examining the evaluation results, we identified a few future directions for this work.

6.1.1 Off-Topic Labeller

We believe that curating a better training dataset for the off-topic classifier would likely improve the effectiveness of this labelling module. As previously explained, the sentences

in the training data were manually filtered to remove greeting comments from StackOverflow data, and non-greeting and non-praising comments from the YouTube data. Although we could try to gather similar data from bug reports, we believe that investigating NLP word embedding techniques to further improve the accuracy of the off-topic classifier may be a more feasible option.

6.1.2 Description Labeller

After implementing the description labeller, we found a study on automatically detecting missing *steps-to-reproduce* and *expected behaviour* in bug report descriptions [15]. Their approach encoded 154 discourse patterns to capture *steps-to-reproduce*, *expected behaviour*, and *observed behaviour*. An investigation of the use of their approach to improve the `Description Labeller` would be a logical next step in this work.

6.1.3 Plan Labeller

As previously mentioned, our work did not include a module to capture sentences with the *Plan* intention in bug reports. After examining Chaparro et al.'s work, we believe that finding appropriate discourse patterns could be a good starting point.

6.1.4 User Study of Customizable Bug Report Summaries

In this work, we evaluated the effectiveness of the labelling modules using a laboratory experiment. Conducting a user study to determine how useful our automated sentence annotations are to software developers is needed to further validate our approach. The user study would involve software project members with different roles to understand the importance of each label to their tasks and the usefulness of the interface presented in Chapter 4.3.

Bibliography

- [1] Apache tomcat bug reporting. https://tomcat.apache.org/bugreport.html#How_to_write_a_bug_report. Accessed: 2020-02-03.
- [2] Bug report writing guidelines - mozilla. https://developer.mozilla.org/en-US/docs/Mozilla/QA/Bug_writing_guidelines. Accessed: 2020-02-03.
- [3] John Anvik, Lyndon Hiew, and Gail C. Murphy. Coping with an open bug repository. In Margaret-Anne D. Storey, Michael G. Burke, Li-Te Cheng, and André van der Hoek, editors, *Proceedings of the 2005 OOPSLA workshop on Eclipse Technology eXchange, ETX 2005, San Diego, California, USA, October 16-17, 2005*, pages 35–39. ACM, 2005.
- [4] John Anvik, Lyndon Hiew, and Gail C. Murphy. Who should fix this bug? In *Proceedings of the 28th International Conference on Software Engineering, ICSE '06*, pages 361–370, New York, NY, USA, 2006. ACM.
- [5] John Anvik and Gail C. Murphy. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. *ACM Trans. Softw. Eng. Methodol.*, 20(3):10:1–10:35, August 2011.
- [6] Dane Bertram, Amy Volda, Saul Greenberg, and Robert Walker. Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams. In Kori Inkpen, Carl Gutwin, and John C. Tang, editors, *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, CSCW 2010, Savannah, Georgia, USA, February 6-10, 2010*, pages 291–300. ACM, 2010.
- [7] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. What makes a good bug report? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, SIGSOFT '08/FSE-16*, page 308–318, New York, NY, USA, 2008. Association for Computing Machinery.
- [8] Nicolas Bettenburg, Rahul Premraj, Thomas Zimmermann, and Sunghun Kim. Extracting structural information from bug reports. In Ahmed E. Hassan, Michele Lanza, and Michael W. Godfrey, editors, *Proceedings of the 2008 International Working Conference on Mining Software Repositories, MSR 2008 (Co-located with ICSE), Leipzig, Germany, May 10-11, 2008, Proceedings*, pages 27–30. ACM, 2008.
- [9] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

- [10] Gerald Bortis and André van der Hoek. Porchlight: a tag-based approach to bug triaging. In David Notkin, Betty H. C. Cheng, and Klaus Pohl, editors, *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, pages 342–351. IEEE Computer Society, 2013.
- [11] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Networks*, 30(1-7):107–117, 1998.
- [12] Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In W. Bruce Croft, Alistair Moffat, C. J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 335–336. ACM, 1998.
- [13] Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. Summarizing email conversations with clue words. In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, page 91–100, New York, NY, USA, 2007. Association for Computing Machinery.
- [14] Oscar Chaparro, Juan Manuel Florez, Unnati Singh, and Andrian Marcus. Reformulating queries for duplicate bug report detection. In Xinyu Wang, David Lo, and Emad Shihab, editors, *26th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2019, Hangzhou, China, February 24-27, 2019*, pages 218–229. IEEE, 2019.
- [15] Oscar Chaparro, Jing Lu, Fiorella Zampetti, Laura Moreno, Massimiliano Di Penta, Andrian Marcus, Gabriele Bavota, and Vincent Ng. Detecting missing information in bug descriptions. In Eric Bodden, Wilhelm Schäfer, Arie van Deursen, and Andrea Zisman, editors, *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, pages 396–407. ACM, 2017.
- [16] Shaiful Alam Chowdhury and Abram Hindle. Mining stackoverflow to filter out off-topic IRC discussion. In Massimiliano Di Penta, Martin Pinzger, and Romain Robbes, editors, *12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015*, pages 422–425. IEEE Computer Society, 2015.
- [17] Davor Cubranic, Gail C. Murphy, Janice Singer, and Kellogg S. Booth. Hipikat: A project memory for software development. *IEEE Trans. Software Eng.*, 31(6):446–465, 2005.
- [18] Akalanka Galappaththi and John Anvik. Feature evaluation for automatic bug report summarization (S). In Angelo Perkusich, editor, *The 31st International Conference on Software Engineering and Knowledge Engineering, SEKE 2019, Hotel Tivoli, Lisbon, Portugal, July 10-12, 2019*, pages 205–274. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2019.

- [19] Zhongxian Gu, Earl T. Barr, Drew Schleck, and Zhendong Su. Reusing debugging knowledge via trace-based bug search. In Gary T. Leavens and Matthew B. Dwyer, editors, *Proceedings of the 27th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2012, part of SPLASH 2012, Tucson, AZ, USA, October 21-25, 2012*, pages 927–942. ACM, 2012.
- [20] Zhongxian Gu, Earl T. Barr, Drew Schleck, and Zhendong Su. Reusing debugging knowledge via trace-based bug search. In Gary T. Leavens and Matthew B. Dwyer, editors, *Proceedings of the 27th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2012, part of SPLASH 2012, Tucson, AZ, USA, October 21-25, 2012*, pages 927–942. ACM, 2012.
- [21] Udo Hahn and Inderjeet Mani. The challenges of automatic summarization. *IEEE Computer*, 33(11):29–36, 2000.
- [22] Björn Hartmann, Daniel MacDougall, Joel Brandt, and Scott R. Klemmer. What would other programmers do: suggesting solutions to error messages. In Elizabeth D. Mynatt, Don Schoner, Geraldine Fitzpatrick, Scott E. Hudson, W. Keith Edwards, and Tom Rodden, editors, *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Atlanta, Georgia, USA, April 10-15, 2010*, pages 1019–1028. ACM, 2010.
- [23] Abram Hindle and Curtis Onuczko. Preventing duplicate bug reports by continuously querying bug reports. *Empirical Software Engineering*, 24(2):902–936, 2019.
- [24] Beibei Huai, Wenbo Li, Qiansheng Wu, and Meiling Wang. Mining intentions to improve bug report summarization. In Óscar Mortágua Pereira, editor, *The 30th International Conference on Software Engineering and Knowledge Engineering, Hotel Pullman, Redwood City, California, USA, July 1-3, 2018*, pages 320–319. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2018.
- [25] He Jiang, Najam Nazar, Jingxuan Zhang, Tao Zhang, and Zhilei Ren. PRST: A pagerank-based summarization technique for summarizing bug reports with duplicates. *International Journal of Software Engineering and Knowledge Engineering*, 27(6):869–896, 2017.
- [26] Jaweria Kanwal and Onaiza Maqbool. Bug prioritization to facilitate bug report triage. *J. Comput. Sci. Technol.*, 27(2):397–412, 2012.
- [27] A. Kiani-B, M. . Akbarzadeh-T., and M. H. Moeinzadeh. Intelligent extractive text summarization using fuzzy inference systems. In *2006 IEEE International Conference on Engineering of Intelligent Systems*, pages 1–4, April 2006.
- [28] Farshad Kiyoumars. Evaluation of automatic text summarizations based on human summaries. volume 192, pages 3–91, 2015. The Proceedings of 2nd Global Conference on Conference on Linguistics and Foreign Language Teaching.

- [29] Xiaochen Li, He Jiang, Dong Liu, Zhilei Ren, and Ge Li. Unsupervised deep bug report summarization. In Foutse Khomh, Chanchal K. Roy, and Janet Siegmund, editors, *Proceedings of the 26th Conference on Program Comprehension, ICPC 2018, Gothenburg, Sweden, May 27-28, 2018*, pages 144–155. ACM, 2018.
- [30] Rafael Lotufo, Zeeshan Malik, and Krzysztof Czarnecki. Modelling the ‘hurried’ bug report reading process to summarize bug reports. *Empirical Softw. Engg.*, 20(2):516–548, April 2015.
- [31] Senthil Mani, Rose Catherine, Vibha Singhal Sinha, and Avinava Dubey. Ausum: Approach for unsupervised bug report summarization. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE ’12*, pages 11:1–11:11, New York, NY, USA, 2012. ACM.
- [32] Qiaozhu Mei, Jian Guo, and Dragomir R. Radev. Divrank: the interplay of prestige and diversity in information networks. In Bharat Rao, Balaji Krishnapuram, Andrew Tomkins, and Qiang Yang, editors, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 1009–1018. ACM, 2010.
- [33] Gabriel Murray and Giuseppe Carenini. Summarizing spoken and written conversations. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 773–782. ACL, 2008.
- [34] Mir Tafseer Nayeem. Methods of sentence extraction, abstraction and ordering for automatic text summarization. Master’s thesis, Universtiy of Lethbridge, Department of Mathematics and Computer Science, Lethbridge, AB, Canada, 2017.
- [35] Ani Nenkova and Kathleen McKeown. *A Survey of Text Summarization Techniques*. Springer US, Boston, MA, 2012.
- [36] Ani Nenkova, Kathleen McKeown, et al. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233, 2011.
- [37] Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4(2):4–es, May 2007.
- [38] Sarah Rastkar, Gail C. Murphy, and Gabriel Murray. Summarizing software artifacts: A case study of bug reports. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE ’10*, page 505–514, New York, NY, USA, 2010. Association for Computing Machinery.
- [39] Sarah Rastkar, Gail C. Murphy, and Gabriel Murray. Automatic summarization of bug reports. *IEEE Trans. Softw. Eng.*, 40(4):366–380, April 2014.

- [40] Simon Tatham. How to report bugs effectively. <https://www.chiark.greenend.org.uk/~sgtatham/bugs.html>. Accessed: 2020-02-03.
- [41] Ferdian Thung, Pavneet Singh Kochhar, and David Lo. Dupfinder: integrated tool support for duplicate bug report detection. In Ivica Crnkovic, Marsha Chechik, and Paul Grünbacher, editors, *ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden - September 15 - 19, 2014*, pages 871–874. ACM, 2014.
- [42] W.M. Wang, Z. Li, J.W. Wang, and Z.H. Zheng. How far we can go with extractive text summarization? heuristic methods to obtain near upper bounds. *Expert Systems with Applications*, 90:439–463, 2017.
- [43] Geunseok Yang, Tao Zhang, and Byungjeong Lee. Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports. In *IEEE 38th Annual Computer Software and Applications Conference, COMPSAC 2014, Vasteras, Sweden, July 21-25, 2014*, pages 97–106. IEEE Computer Society, 2014.
- [44] Shamima Yeasmin, Chanchal Kumar Roy, and Kevin A. Schneider. Interactive visualization of bug reports using topic evolution and extractive summaries. In *30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014*, pages 421–425. IEEE Computer Society, 2014.
- [45] Tao Zhang, Geunseok Yang, Byungjeong Lee, and Alvin T. S. Chan. Guiding bug triage through developer analysis in bug reports. *International Journal of Software Engineering and Knowledge Engineering*, 26(3):405–432, 2016.
- [46] Xiaojin Zhu, Andrew B. Goldberg, Jurgen Van Gael, and David Andrzejewski. Improving diversity in ranking using absorbing random walks. In Candace L. Sidner, Tanja Schultz, Matthew Stone, and ChengXiang Zhai, editors, *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, April 22-27, 2007, Rochester, New York, USA*, pages 97–104. The Association for Computational Linguistics, 2007.

Appendix A

Annotated Bug Reports

We derived nine labels to identify the content found in bug reports. We asked annotators to fill a comma separated file (CSV) to indicate the contents of a sentence except for the label *Topic*. Tables A.1 - A.36 show the annotated bug reports.

Table A.1: Annotated bug report (*Firefox:#495584*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	0	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	1	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0
1.12	1	0	0	0	0	0	0	0
1.13	1	0	0	0	0	0	0	0
2.1	0	0	0	0	1	0	0	0
2.2	0	0	0	0	1	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0
3.1	0	0	0	0	1	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	1	0	0	0
3.4	0	0	0	0	0	0	0	0
4.1	0	1	0	0	1	0	0	0
4.2	0	1	0	1	0	0	0	0

5.1	0	0	0	0	0	0	0	0
5.2	0	0	1	0	0	0	0	0
5.3	0	0	1	0	0	0	0	0
5.4	0	0	0	0	0	1	0	0
5.5	0	0	0	1	0	0	0	0
5.6	0	0	0	1	0	0	0	0
6.1	0	0	0	0	1	0	1	0

Table A.2: Annotated bug report (*Firefox:#449596*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
2.1	0	0	0	0	1	0	0	0
2.2	0	0	0	0	0	0	0	0
3.1	0	0	0	0	1	0	0	0
4.1	0	0	0	0	1	0	0	0
4.2	0	0	0	0	0	0	0	0
5.1	0	0	0	0	1	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	1	0	1	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	1	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	1	0	0	0	0	0
6.5	0	1	0	1	0	0	0	0
6.6	0	1	0	1	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	1	0	0	0	0	0
7.3	0	0	0	0	0	1	0	0
7.4	0	0	1	0	0	0	0	0
7.5	0	0	0	1	0	0	0	0
7.6	0	0	0	0	0	0	0	0
8.1	0	0	0	0	1	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0

8.4	0	0	0	0	0	0	0	1
8.5	0	0	0	0	0	0	0	1
8.6	0	0	0	0	0	0	0	1
8.7	0	0	0	0	0	0	0	1
8.8	0	1	0	0	0	0	0	0
8.9	0	0	0	0	0	0	0	1
9.1	0	0	0	0	0	0	0	0
9.2	0	0	1	0	0	0	0	0
9.3	0	0	0	0	0	0	0	0
9.4	0	0	0	0	0	1	0	0
10.1	0	0	0	0	1	0	0	0
11.1	0	0	0	0	1	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	0	0	0	0	0	0	0
11.4	0	0	0	0	0	0	0	0
11.5	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	1	0	0
13.2	0	0	0	0	0	0	0	0
13.3	0	1	0	0	0	0	0	0
13.4	0	0	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0
14.2	0	0	1	0	0	0	0	0
14.3	0	0	0	0	0	0	0	0
14.4	0	0	0	0	0	0	0	0
14.5	0	0	0	0	0	0	0	0
14.6	0	0	0	0	0	0	0	0
14.7	0	0	0	0	0	0	0	0

Table A.3: Annotated bug report (*Firefox:#491925*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0

2.2	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	1	0	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0
5.6	0	0	0	0	0	0	0	0
5.7	0	0	0	0	0	0	0	0
5.8	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	1
9.3	0	0	0	0	0	0	0	1
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
11.1	0	1	0	1	0	0	0	0
12.1	0	0	0	0	0	0	0	0
12.2	0	0	1	0	0	0	0	0
12.3	0	0	0	1	0	0	0	0
12.4	0	0	0	0	0	0	0	0
12.5	0	0	0	0	0	0	0	0
12.6	0	0	0	1	0	0	0	0
12.7	0	0	0	1	0	0	0	0
12.8	0	0	0	1	0	0	0	0
13.1	0	1	0	1	0	0	0	0
13.2	0	1	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0
14.2	0	0	1	0	0	0	0	0

14.3	0	0	0	0	0	0	0	0
14.4	0	0	0	1	0	0	0	0
14.5	0	0	0	1	0	0	0	0
14.6	0	0	0	0	0	0	0	0
14.7	0	0	0	1	0	0	0	0
15.1	0	0	0	0	0	0	0	0
15.2	0	0	1	0	0	0	0	0
15.3	0	0	0	0	0	0	0	0
15.4	0	0	1	0	0	0	0	0
15.5	0	0	0	1	0	0	0	0
15.6	0	0	0	0	0	0	0	0
15.7	0	0	0	1	0	0	0	0
16.1	0	0	0	0	0	0	0	0
16.2	0	0	0	0	0	0	0	0
16.3	0	0	0	0	0	1	0	0
17.1	0	0	0	0	1	0	0	0
17.2	0	0	0	0	0	0	0	0
17.3	0	0	0	0	0	0	0	0
17.4	0	0	0	0	0	0	0	0
17.5	0	0	0	0	0	0	0	0
17.6	0	0	0	0	0	0	0	0
17.7	0	0	0	0	0	0	0	0
17.8	0	0	0	0	0	0	0	0
17.9	0	0	0	0	0	0	0	0
18.1	0	0	0	0	1	0	0	0
19.1	0	0	0	0	1	0	0	0
19.2	0	0	0	0	0	0	0	0
19.3	0	0	0	0	0	0	0	0
20.1	0	0	0	0	1	0	0	0
20.2	0	0	0	0	0	0	0	1
21.1	0	0	0	0	0	0	1	0
21.2	0	0	0	0	0	0	0	0
22.1	0	0	0	0	0	0	1	0
23.1	0	0	0	0	1	0	0	0
23.2	0	0	0	0	0	0	0	1
23.3	0	0	0	0	0	0	0	1
24.1	0	0	0	0	0	0	0	0
24.2	0	0	0	0	0	0	0	0
24.3	0	0	0	0	0	0	0	0
24.4	0	0	0	0	0	0	0	0
24.5	0	0	0	0	0	0	0	0

25.1	0	0	0	0	0	0	1	0
------	---	---	---	---	---	---	---	---

Table A.4: Annotated bug report (*Eclipse:#250125*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0
1.7	0	0	0	0	0	0	0	1
1.8	0	0	0	0	0	0	0	1
1.9	0	0	0	0	0	0	0	1
1.10	0	0	0	0	0	0	0	1
1.11	0	0	0	0	0	0	0	1
1.12	0	0	0	0	0	0	0	1
1.13	1	0	0	0	0	0	0	0
1.14	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	1	0	1	0	0	0	0
3.2	0	0	0	0	0	0	0	0
4.1	0	0	1	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	1	0	0	0	0
4.4	0	0	0	0	0	0	0	0
4.5	0	0	0	1	0	0	0	0
4.6	0	0	0	1	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
7.1	0	0	0	0	1	0	0	0
7.2	0	0	0	0	1	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	0	0	0
7.5	0	0	0	0	0	0	0	0
7.6	0	0	0	0	1	0	0	0
7.7	0	0	0	0	0	0	0	0
7.8	0	0	0	0	0	0	0	0

7.9	0	1	0	1	0	0	0	0
7.10	0	0	0	0	0	0	0	0
7.11	0	0	0	0	0	0	0	0
8.1	0	0	1	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	1	0	0	0	0
8.4	0	0	0	1	0	0	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	0

Table A.5: Annotated bug report (*Eclipse:#224588*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	1
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	0	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	0	0	0	0	0	1	0	0
1.11	0	0	0	0	0	1	0	0
1.12	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	1	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	1
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	1
3.8	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
4.6	0	0	0	0	0	0	0	0

5.1	0	0	0	0	1	0	0	0
5.2	0	0	0	0	1	0	0	0
5.3	0	0	0	0	1	0	0	0
5.4	0	0	0	0	0	0	0	0
5.5	0	1	0	0	0	0	0	0
5.6	0	0	0	0	0	0	0	0
6.1	0	0	0	0	1	0	0	0
6.2	0	0	0	0	0	0	0	1
7.1	0	0	0	0	0	0	0	0
7.2	0	0	1	0	0	0	0	0
7.3	0	1	0	1	0	0	0	0
7.4	0	1	0	1	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	1	0	0	0	0	0
8.3	0	0	1	0	0	0	0	0
8.4	0	0	1	0	0	0	0	0
8.5	0	0	0	1	0	0	0	0
8.6	0	0	0	0	0	0	0	0
9.1	0	0	0	0	1	0	0	0
9.2	0	0	0	0	1	0	0	0
10.1	0	0	0	0	1	0	0	0
10.2	0	0	0	0	0	0	0	1
11.1	0	0	0	0	1	0	0	0
11.2	0	0	0	0	1	0	0	0
11.3	0	0	0	0	1	0	0	0
11.4	0	0	0	0	0	0	0	0
12.1	0	0	0	0	1	0	0	0
12.2	0	0	0	0	1	0	0	0
12.3	0	0	0	0	1	0	0	0
13.1	0	0	0	0	1	0	0	0
13.2	0	0	0	0	0	0	0	0
13.3	0	0	0	0	0	0	0	0
14.1	0	0	0	0	1	0	0	0
14.2	0	0	0	0	0	0	0	0
14.3	0	0	0	0	0	0	0	0
15.1	0	0	0	0	1	0	0	0
15.2	0	0	0	0	0	0	0	0
15.3	0	0	0	0	0	0	0	0
15.4	0	0	0	0	0	0	0	0
15.5	0	0	0	0	0	0	0	0
16.1	0	0	0	0	0	0	0	0

16.2	0	0	0	0	0	0	0	0
16.3	0	0	0	0	0	0	0	0
16.4	0	0	0	0	0	0	0	0
16.5	0	0	0	0	0	0	0	0
16.6	0	0	0	0	0	0	0	0
16.7	0	0	0	0	0	0	0	0
16.8	0	0	0	0	0	0	0	1
16.9	0	0	0	0	0	0	0	1
16.10	0	0	0	0	0	0	0	1
16.11	0	0	0	0	0	0	0	0
17.1	0	0	0	0	0	0	0	0
17.2	0	0	0	0	1	0	0	0
17.3	0	0	0	0	0	0	0	0
17.4	0	0	0	0	0	0	0	0
17.5	0	0	0	0	1	0	0	0
17.6	0	0	0	0	1	0	0	0
17.7	0	0	0	0	0	0	0	0
17.8	0	0	0	0	0	0	0	0
17.9	0	0	0	0	0	0	0	0
17.10	0	0	0	0	0	0	0	0
17.11	0	0	0	0	0	0	0	0
17.12	0	0	0	0	0	0	0	0
17.13	0	0	0	0	1	0	0	0
17.14	0	0	0	0	0	0	0	0
17.15	0	0	0	0	0	0	0	0
17.16	0	0	0	0	0	0	0	0
17.17	0	0	0	0	0	1	0	0
17.18	0	0	0	0	0	1	0	0
18.1	0	0	0	0	0	0	0	0
18.2	0	0	0	0	0	0	0	0
19.1	0	0	0	0	0	1	0	0
19.2	0	0	0	0	0	0	0	0
19.3	0	0	0	0	0	0	0	1
19.4	0	0	0	0	0	0	0	1

Table A.6: Annotated bug report (*Eclipse:#223734*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0

1.3	0	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0
2.1	0	1	0	1	0	0	0	0
2.2	0	1	0	0	0	0	0	0
2.3	0	1	0	0	0	0	0	0
2.4	0	1	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	1	0	0	0	0	0
4.3	0	0	1	0	0	0	0	0
4.4	0	0	1	0	0	0	0	0
4.5	0	0	1	0	0	0	0	0
4.6	0	0	0	0	0	0	0	0
4.7	0	0	0	0	0	0	0	0
4.8	0	0	0	1	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	1
7.3	0	0	0	0	0	0	0	1
7.4	0	0	0	0	0	0	0	1
8.1	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	0	0	0

Table A.7: Annotated bug report (*Firefox:#437797*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	1	0	0
3.1	0	0	0	0	0	0	0	0

3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
6.1	0	0	0	0	1	0	0	0
6.2	0	0	0	0	1	0	0	0
6.3	0	0	0	0	1	0	0	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	0	0	0	0	0	0
7.1	0	1	0	1	0	0	0	0
7.2	0	0	0	0	0	0	1	0
7.3	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	1	0	0	0	0	0
8.3	0	0	0	1	0	0	0	0
8.4	0	0	0	0	0	0	0	0
9.1	0	1	0	0	0	0	0	0
9.2	0	1	0	1	0	0	0	0
9.3	0	1	0	1	0	0	0	0
9.4	0	0	0	0	0	1	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	1	0	0	0	0	0
10.3	0	0	0	0	0	0	0	0
10.4	0	0	1	0	0	0	0	0
10.5	0	0	1	0	0	0	0	0
10.6	0	0	0	0	0	1	0	0
10.7	0	0	0	1	0	0	0	0
10.8	0	0	0	1	0	0	0	0
10.9	0	0	0	1	0	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	0	0	0
12.2	0	0	0	0	0	0	0	0
12.3	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
13.2	0	0	0	0	0	0	0	0

14.1	0	0	0	0	0	0	0	0
14.2	0	0	0	0	0	0	0	0
14.3	0	0	0	0	0	1	0	0
15.1	0	0	0	0	0	0	0	0
16.1	0	0	0	0	1	0	0	0
16.2	0	0	0	0	0	0	0	0
16.3	0	0	0	0	0	0	0	0
17.1	0	0	0	0	0	0	1	0

Table A.8: Annotated bug report (*Eclipse:#260502*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	0	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	0	0	0	0	0	0	0	0
1.10	0	0	0	0	0	0	0	1
1.11	0	0	0	0	0	0	0	1
1.12	0	0	0	0	0	0	0	1
1.13	0	0	0	0	0	0	0	1
1.14	0	0	0	0	0	0	0	0
1.15	1	0	0	0	0	0	0	0
1.16	0	0	0	0	0	0	0	1
1.17	0	0	0	0	0	0	0	1
1.18	0	0	0	0	0	0	0	1
1.19	0	0	0	0	0	0	0	1
1.20	0	0	0	0	0	0	0	1
1.21	0	0	0	0	0	0	0	1
1.22	0	0	0	0	0	0	0	1
1.23	0	0	0	0	0	0	0	1
1.24	0	0	0	0	0	0	0	1
1.25	0	0	0	0	0	0	0	1
1.26	1	0	0	0	0	0	0	0
1.27	0	0	0	0	0	0	0	0
1.28	1	0	0	0	0	0	0	0

1.29	0	0	0	0	0	0	0	0
1.30	0	0	0	0	0	0	0	0
1.31	0	0	0	0	0	0	0	0
1.32	0	0	0	0	0	0	0	0
1.33	0	0	0	0	0	0	0	0
1.34	0	0	0	0	0	0	0	0
1.35	1	0	0	0	0	0	0	0
1.36	1	0	0	0	0	0	0	0
1.37	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0
2.6	0	0	0	0	0	0	0	0
2.7	0	0	0	0	0	0	0	0
2.8	0	0	0	0	0	0	0	0
2.9	0	0	0	0	0	0	0	0
2.10	0	0	0	0	0	0	0	0
2.11	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	1	0	0
3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
4.6	0	0	0	0	0	0	0	0
4.7	0	0	0	0	0	0	0	0
4.8	0	0	0	0	0	0	0	0
4.9	0	0	0	0	0	0	0	0
4.10	0	0	0	0	0	0	0	0
4.11	0	0	0	0	0	0	0	0
5.1	0	0	0	0	1	0	0	0
5.2	0	0	0	0	1	0	0	0
6.1	0	0	0	0	1	0	0	0

6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
7.1	0	0	0	0	1	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	1	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	0	0
8.6	0	0	0	0	0	0	0	0
8.7	0	0	0	0	0	0	0	0
8.8	0	0	0	0	0	1	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	1	0	0

Table A.9: Annotated bug report (*Firefox:#328600*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	1
1.2	0	0	0	0	0	0	0	1
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	1
5.2	0	0	0	0	0	0	0	1
5.3	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0

Table A.10: Annotated bug report (*GIMP:#156905*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0

1.2	1	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	0	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
5.1	0	0	0	0	1	0	0	0
6.1	0	0	0	0	1	0	0	0
7.1	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
10.1	0	0	0	0	1	0	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	0	0	0
10.4	0	0	0	0	0	0	0	1
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	0	0	0
12.2	0	0	0	0	0	0	0	0
12.3	0	0	0	0	0	0	0	0
12.4	0	0	0	0	0	0	0	0
12.5	0	0	0	0	0	0	0	1
12.6	0	0	0	0	0	0	0	0
13.1	0	0	0	0	1	0	0	0
13.2	0	0	0	0	0	0	0	0
13.3	0	0	0	0	1	0	0	0
13.4	0	0	0	0	0	0	0	0
13.5	0	0	0	0	1	0	0	0
14.1	0	0	0	0	0	0	0	0

15.1	0	0	0	0	1	0	0	0
15.2	0	0	0	0	1	0	0	0
15.3	0	0	0	0	0	0	0	0
16.1	0	0	0	0	0	0	0	0
16.2	0	0	0	0	0	0	0	1
16.3	0	0	0	0	1	0	0	0
16.4	0	0	0	0	0	0	0	0
16.5	0	0	0	0	0	0	0	0

Table A.11: Annotated bug report (*GIMP:#164995*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
4.1	0	1	0	0	0	1	0	0
4.2	0	1	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	0	0	0	0	0	0
6.6	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	0	0	0
7.5	0	0	0	0	0	0	0	0
7.6	0	0	0	0	0	0	0	0
7.7	0	0	0	0	0	0	0	0
8.1	0	0	1	0	0	0	0	0
8.2	0	0	0	0	0	1	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	1	0
9.1	0	0	0	0	0	0	0	0

9.2	0	0	0	0	0	0	0	0
9.3	0	0	0	0	1	0	0	0
9.4	0	0	0	0	0	0	0	0
9.5	0	0	0	0	0	0	0	0
9.6	0	0	0	0	0	0	0	0
9.7	0	0	0	0	1	0	0	0
9.8	0	0	0	0	0	0	0	1
9.9	0	0	0	0	0	0	0	1
9.10	0	0	0	0	0	0	0	1
9.11	0	0	0	0	0	0	0	1
9.12	0	0	0	0	0	0	0	1
9.13	0	0	0	0	0	0	0	0
9.14	0	0	0	0	0	0	0	1

Table A.12: Annotated bug report (*GIMP:#170801*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	1	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0
1.12	0	0	0	0	0	0	0	0
1.13	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0
3.1	0	0	0	0	1	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	1	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0

3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	0
3.8	0	0	0	0	0	0	0	0
3.9	0	0	0	0	0	0	0	0
3.10	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	1	0	0	0
4.6	0	0	0	0	0	0	0	0
4.7	0	0	0	0	1	0	0	0
4.8	0	0	0	0	0	0	0	0
4.9	0	0	0	0	0	0	0	1
4.10	0	0	0	0	1	0	0	0
4.11	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	1	0	0
5.2	0	0	0	0	0	1	0	0
5.3	0	0	0	0	0	0	0	0
6.1	0	0	0	0	1	0	0	0
6.2	0	0	0	0	0	0	0	0
7.1	0	0	0	0	1	0	0	0
7.2	0	0	0	0	0	0	0	1
7.3	0	0	0	0	0	0	0	1
7.4	0	0	0	0	0	0	0	1
7.5	0	0	0	0	0	0	0	1
7.6	0	0	0	0	0	0	0	1
7.7	0	0	0	0	0	0	0	1
7.8	0	0	0	0	0	0	0	1
7.9	0	0	0	0	0	0	0	1
7.10	0	0	0	0	0	0	0	1
8.1	0	0	0	0	0	1	0	0
8.2	0	0	0	0	0	1	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	1	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	1	0	1	0	0	0	0

12.1	0	0	1	0	0	0	0	0
12.2	0	0	0	1	0	0	0	0
12.3	0	1	0	1	0	0	0	0
12.4	0	0	0	0	0	0	0	0
13.1	0	0	1	0	0	0	0	0
13.2	0	0	0	1	0	0	0	0
13.3	0	0	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0
14.2	0	0	0	0	0	0	0	0

Table A.13: Annotated bug report (*GIMP:#364852*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0
1.7	0	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	1	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0
1.12	1	0	0	0	0	0	0	0
1.13	1	0	0	0	0	0	0	0
1.14	1	0	0	0	0	0	0	0
1.15	1	0	0	0	0	0	0	0
1.16	1	0	0	0	0	0	0	0
1.17	1	0	0	0	0	0	0	0
1.18	0	0	0	0	0	0	0	0
1.19	0	0	0	0	0	0	0	0
1.20	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0

3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
4.6	0	0	0	0	0	0	0	0
4.7	0	0	0	0	0	0	0	0
4.8	0	0	0	0	0	0	0	0
4.9	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	0	0	0	0	0	0
6.6	0	0	0	0	0	0	0	0
6.7	0	0	0	0	0	0	0	1
6.8	0	0	0	0	0	0	0	1
6.9	0	0	0	0	0	0	0	1
6.10	0	0	0	0	1	0	0	0
6.11	0	0	0	0	0	0	0	1

Table A.14: Annotated bug report (*GNUCash:#168803*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	1	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0
1.12	1	0	0	0	0	0	0	0
1.13	0	0	0	0	0	0	0	0

1.14	0	0	0	0	0	0	0	0
1.15	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	1	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	1	0	0
4.1	0	1	0	0	0	0	0	0
4.2	0	1	0	1	0	0	0	0
4.3	0	1	0	0	0	0	0	0
4.4	0	1	0	0	0	0	0	0
4.5	0	0	0	0	0	1	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	1	0	0	0	0	0
5.3	0	0	1	0	0	0	0	0
5.4	0	0	1	0	0	0	0	0
5.5	0	0	1	0	0	0	0	0
5.6	0	0	0	0	0	0	0	0
5.7	0	0	0	0	0	0	0	0
5.8	0	0	0	1	0	0	0	0
5.9	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	0	0
8.6	0	0	0	0	0	0	0	0
8.7	0	0	0	0	1	0	0	0
8.8	0	0	0	0	0	0	0	0
8.9	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0

Table A.15: Annotated bug report (*gvfs:#522933*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	1	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	1	0	0	0
7.5	0	0	0	0	1	0	0	0
8.1	0	0	0	0	1	0	0	0
8.2	0	0	0	0	1	0	0	0
8.3	0	0	0	0	1	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	0	0
8.6	0	0	0	0	1	0	0	0
9.1	0	0	0	0	1	0	0	0
9.2	0	0	0	0	0	0	0	0
9.3	0	0	0	0	0	0	0	0
9.4	0	0	0	0	0	0	0	0
9.5	0	0	0	0	0	0	0	1
9.6	0	0	0	0	0	0	0	1
9.7	0	0	0	0	0	0	0	1
9.8	0	0	0	0	0	0	0	1
9.9	0	0	0	0	0	0	0	1
9.10	0	0	0	0	0	0	0	1
9.11	0	0	0	0	0	0	0	1
9.12	0	0	0	0	0	0	0	1
9.13	0	0	0	0	0	0	0	1
9.14	0	0	0	0	0	0	0	1
9.15	0	0	0	0	0	0	0	1
9.16	0	0	0	0	0	0	0	1
9.17	0	0	0	0	0	0	0	1

9.18	0	0	0	0	0	0	0	0
9.19	0	0	0	0	0	0	0	0
9.20	0	0	0	0	0	0	0	1
9.21	0	0	0	0	0	0	0	1
9.22	0	0	0	0	0	0	0	1
9.23	0	0	0	0	0	0	0	1
9.24	0	0	0	0	0	0	0	1
9.25	0	0	0	0	0	0	0	1
9.26	0	0	0	0	0	0	0	1
9.27	0	0	0	0	0	0	0	1
9.28	0	0	0	0	0	0	0	1
9.29	0	0	0	0	0	0	0	0
9.30	0	0	0	0	0	0	0	0
9.31	0	0	0	0	0	0	0	1
9.32	0	0	0	0	0	0	0	1
9.33	0	0	0	0	0	0	0	1
9.34	0	0	0	0	0	0	0	1
9.35	0	0	0	0	0	0	0	1
9.36	0	0	0	0	0	0	0	1
9.37	0	0	0	0	0	0	0	1
9.38	0	0	0	0	0	0	0	1
9.39	0	0	0	0	0	0	0	1
9.40	0	0	0	0	0	0	0	1
9.41	0	0	0	0	0	0	0	1
9.42	0	0	0	0	0	0	0	1
9.43	0	0	0	0	0	0	0	1
9.44	0	0	0	0	0	0	0	1
9.45	0	0	0	0	0	0	0	1
9.46	0	0	0	0	0	0	0	1
9.47	0	0	0	0	0	0	0	1
9.48	0	0	0	0	0	0	0	1
9.49	0	0	0	0	0	0	0	1
9.50	0	0	0	0	0	0	0	1
9.51	0	0	0	0	0	0	0	1
9.52	0	0	0	0	0	0	0	1
9.53	0	0	0	0	0	0	0	1
9.54	0	0	0	0	0	0	0	1
9.55	0	0	0	0	0	0	0	1
9.56	0	0	0	0	0	0	0	1
9.57	0	0	0	0	0	0	0	1
9.58	0	0	0	0	0	0	0	1

9.59	0	0	0	0	0	0	0	1
9.60	0	0	0	0	0	0	0	1
9.61	0	0	0	0	0	0	0	0
9.62	0	0	0	0	0	0	0	0
9.63	0	0	0	0	0	0	0	0
9.64	0	0	0	0	0	0	0	0
9.65	0	0	0	0	0	0	0	0
9.66	0	0	0	0	0	0	0	0
9.67	0	0	0	0	0	0	0	0
9.68	0	0	0	0	0	0	0	0
9.69	0	0	0	0	1	0	0	0
10.1	0	0	0	0	1	0	0	0
10.2	0	0	0	0	1	0	0	0
10.3	0	0	0	0	1	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
12.1	0	0	0	0	1	0	0	0
12.2	0	0	0	0	1	0	0	0
12.3	0	0	0	0	0	0	0	0
12.4	0	0	0	0	0	0	0	0

Table A.16: Annotated bug report (*GTK:#64222*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0
1.7	0	0	0	0	0	0	0	0
1.8	0	0	0	0	0	0	0	1
1.9	0	0	0	0	0	0	0	1
1.10	0	0	0	0	0	0	0	1
1.11	0	0	0	0	0	0	0	1
1.12	0	0	0	0	0	0	0	1
1.13	0	0	0	0	0	0	0	1
1.14	0	0	0	0	0	0	0	1
1.15	0	0	0	0	0	0	0	1
1.16	0	0	0	0	0	0	0	1

1.17	0	0	0	0	0	0	0	1
1.18	0	0	0	0	0	0	0	1
1.19	0	0	0	0	0	0	0	1
1.20	0	0	0	0	0	0	0	1
1.21	0	0	0	0	0	0	0	1
1.22	0	0	0	0	0	0	0	1
1.23	0	0	0	0	0	0	0	1
1.24	0	0	0	0	0	0	0	1
1.25	0	0	0	0	0	0	0	1
1.26	0	0	0	0	0	0	0	1
1.27	0	0	0	0	0	0	0	1
1.28	0	0	0	0	0	0	0	1
1.29	0	0	0	0	0	0	0	1
1.30	0	0	0	0	0	0	0	1
1.31	0	0	0	0	0	0	0	1
1.32	0	0	0	0	0	0	0	1
1.33	0	0	0	0	0	0	0	1
1.34	0	0	0	0	0	0	0	1
1.35	0	0	0	0	0	0	0	1
1.36	0	0	0	0	0	0	0	1
1.37	0	0	0	0	0	0	0	1
1.38	0	0	0	0	0	0	0	1
1.39	0	0	0	0	0	0	0	1
1.40	0	0	0	0	0	0	0	1
1.41	0	0	0	0	0	0	0	1
1.42	0	0	0	0	0	0	0	1
1.43	0	0	0	0	0	0	0	1
1.44	0	0	0	0	0	0	0	1
1.45	0	0	0	0	0	0	0	1
1.46	0	0	0	0	0	0	0	1
1.47	0	0	0	0	0	0	0	1
1.48	0	0	0	0	0	0	0	1
1.49	0	0	0	0	0	0	0	1
1.50	0	0	0	0	0	0	0	1
2.1	0	0	0	0	0	1	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	1
2.4	0	0	0	0	0	0	0	1
2.5	0	0	0	0	0	0	0	1
2.6	0	0	0	0	0	0	0	0
2.7	0	0	0	0	0	0	0	0

2.8	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
4.6	0	0	0	0	0	0	0	0
4.7	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0

Table A.17: Annotated bug report (*GTK:#514396*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	1	0
2.1	0	0	0	0	1	0	0	0
2.2	0	0	0	0	1	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
5.1	0	0	0	0	1	0	0	0
5.2	0	0	0	0	1	0	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
7.1	0	0	0	0	1	0	0	0

8.1	0	0	0	0	1	0	0	0
8.2	0	0	0	0	1	0	0	0
9.1	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	0	0	0
12.2	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
13.2	0	0	0	0	0	0	0	0
13.3	0	0	0	0	0	0	0	0
14.1	0	0	0	0	1	0	0	0
14.2	0	0	0	0	1	0	0	0
14.3	0	0	0	0	0	1	0	0
14.4	0	0	0	0	0	1	0	0
15.1	0	0	0	0	0	0	0	0
16.1	0	0	0	0	0	0	0	0
16.2	0	0	0	0	0	1	0	0
17.1	0	0	0	0	0	0	0	0
17.2	0	0	0	0	0	0	0	0
17.3	0	0	0	0	0	0	0	0
17.4	0	0	0	0	0	0	0	0
17.5	0	0	0	0	0	0	0	0
17.6	0	0	0	0	0	0	0	0
17.7	0	0	0	0	0	0	0	0
17.8	0	0	0	0	0	0	0	0
18.1	0	0	0	0	1	0	0	0
18.2	0	0	0	0	1	0	0	0
18.3	0	0	0	0	0	1	0	0
18.4	0	0	0	0	1	0	0	0
18.5	0	0	0	0	0	0	0	0
18.6	0	0	0	0	0	0	0	0
18.7	0	0	0	0	0	0	0	0
18.8	0	0	0	0	0	0	0	0
18.9	0	0	0	0	0	0	0	0
18.10	0	0	0	0	0	0	0	0
19.1	0	0	0	0	0	0	0	0
19.2	0	0	0	0	0	0	0	0
19.3	0	0	0	0	0	0	0	0

20.1	0	0	0	0	0	0	0	0
20.2	0	0	0	0	0	0	0	0
20.3	0	0	0	0	0	0	0	0
20.4	0	0	0	0	0	0	0	0
20.5	0	0	0	0	0	0	0	0
20.6	0	0	0	0	0	0	0	0
20.7	0	0	0	0	0	0	0	0
21.1	0	0	0	0	0	0	0	0
21.2	0	0	0	0	0	0	0	0
21.3	0	0	0	0	0	0	0	0
22.1	0	0	0	0	1	0	0	0
22.2	0	0	0	0	0	0	0	0
22.3	0	0	0	0	0	0	0	0
22.4	0	0	0	0	0	0	0	1
22.5	0	0	0	0	0	0	0	1
22.6	0	0	0	0	0	0	0	1
22.7	0	0	0	0	0	0	0	1
22.8	0	0	0	0	0	0	0	1
22.9	0	0	0	0	0	0	0	0
22.10	0	0	0	0	1	0	0	0

Table A.18: Annotated bug report (*Eclipse:#215879*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	1
1.5	0	0	0	0	0	0	0	1
1.6	0	0	0	0	0	0	0	1
1.7	0	0	0	0	0	0	0	1
1.8	0	0	0	0	0	0	0	1
1.9	0	0	0	0	0	0	0	1
1.10	0	0	0	0	0	0	0	1
1.11	0	0	0	0	0	0	0	1
1.12	0	0	0	0	0	0	0	1
1.13	0	0	0	0	0	0	0	1
1.14	0	0	0	0	0	0	0	1
1.15	0	0	0	0	0	0	0	1
1.16	1	0	0	0	0	0	0	0

1.17	1	0	0	0	0	0	0	0
1.18	0	0	0	0	0	0	0	0
1.19	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	1
4.3	0	0	0	0	0	0	0	1
4.4	0	0	0	0	0	0	0	1
4.5	0	0	0	0	0	0	0	0
4.6	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	1	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	1	0	0
8.6	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	1	0

Table A.19: Annotated bug report (*Eclipse:#69350*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	1	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0

2.4	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0
2.6	0	0	0	0	0	0	0	0
2.7	0	0	0	0	0	0	0	0
2.8	0	0	0	0	0	0	0	0
2.9	0	0	0	0	0	0	0	0
2.10	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	1	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	0
3.8	0	0	0	0	0	0	0	0
3.9	0	0	0	0	0	0	0	0
3.10	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
5.1	0	0	0	0	1	0	0	0
5.2	0	0	0	0	0	0	0	1
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0
5.6	0	0	0	0	0	0	0	0
5.7	0	0	0	0	0	0	0	0
5.8	0	0	0	0	0	0	0	0
5.9	0	0	0	0	0	0	0	0
6.1	0	0	0	0	1	0	0	0
6.2	0	0	0	0	0	0	0	1
6.3	0	0	0	0	0	0	0	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	0	0
9.1	0	0	0	0	1	0	0	0
9.2	0	0	0	0	0	0	0	1

10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	0	0	0	1	0	0	0
12.1	0	0	0	0	0	0	0	0
12.2	0	0	0	0	0	1	0	0
13.1	0	0	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0

Table A.20: Annotated bug report (*Eclipse:#226688*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	1	0	0
1.2	1	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	0	0	0	0	0	1	0	0
2.1	0	0	0	0	0	1	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0
2.6	0	0	0	0	0	0	0	0
2.7	0	0	0	0	0	0	0	0
2.8	0	0	0	0	0	0	0	0
2.9	0	0	0	0	0	0	0	0
2.10	0	0	0	0	0	0	0	0
2.11	0	0	0	0	0	0	0	0
2.12	0	0	0	0	0	1	0	0
3.1	0	0	0	0	1	0	0	0
3.2	0	0	0	0	0	0	0	0
4.1	0	0	0	0	1	0	0	0
4.2	0	0	0	0	0	0	0	0
5.1	0	0	0	0	1	0	0	0
5.2	0	0	0	0	0	0	0	0

6.1	0	0	0	0	0	1	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	0	0	0	1	0	0
6.5	0	0	0	0	0	1	0	0
7.1	0	0	0	0	0	1	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	0	0	0
7.5	0	0	0	0	0	1	0	0
8.1	0	0	0	0	0	1	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	0	0
8.6	0	0	0	0	0	1	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	0
9.3	0	0	0	0	1	0	0	0

Table A.21: Annotated bug report (*Eclipse:#276131*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
2.1	0	0	0	0	1	0	0	0
3.1	0	0	0	0	1	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
6.1	0	0	0	0	1	0	0	0
6.2	0	0	0	0	1	0	0	0
7.1	0	0	0	0	1	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0

8.2	0	0	0	0	1	0	0	0
-----	---	---	---	---	---	---	---	---

Table A.22: Annotated bug report (*Bugzilla Calendar:#429126*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	1
1.2	0	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	1	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0
1.12	1	0	0	0	0	0	0	0
1.13	1	0	0	0	0	0	0	0
1.14	1	0	0	0	0	0	0	0
1.15	1	0	0	0	0	0	0	0
1.16	1	0	0	0	0	0	0	0
1.17	1	0	0	0	0	0	0	0
1.18	0	0	0	0	0	0	0	0
1.19	0	0	0	0	0	0	0	1
1.20	0	0	0	0	0	0	0	1
1.21	0	0	0	0	0	0	0	1
1.22	0	0	0	0	0	0	0	1
1.23	0	0	0	0	0	0	0	1
1.24	0	0	0	0	0	0	0	1
1.25	0	0	0	0	0	0	0	1
1.26	0	0	0	0	0	0	0	1
1.27	0	0	0	0	0	0	0	1
1.28	0	0	0	0	0	0	0	1
1.29	0	0	0	0	0	0	0	1
1.30	0	0	0	0	0	0	0	1
1.31	0	0	0	0	0	0	0	1
1.32	0	0	0	0	0	0	0	1
1.33	0	0	0	0	0	0	0	1
1.34	0	0	0	0	0	0	0	1
1.35	0	0	0	0	0	0	0	1

1.36	0	0	0	0	0	0	0	1
1.37	0	0	0	0	0	0	0	1
1.38	0	0	0	0	0	0	0	1
1.39	0	0	0	0	0	0	0	1
1.40	0	0	0	0	0	0	0	1
1.41	0	0	0	0	0	0	0	1
1.42	0	0	0	0	0	0	0	1
1.43	0	0	0	0	0	0	0	1
1.44	0	0	0	0	0	0	0	1
1.45	0	0	0	0	0	0	0	1
1.46	0	0	0	0	0	0	0	1
1.47	0	0	0	0	0	0	0	1
1.48	0	0	0	0	0	0	0	1
1.49	0	0	0	0	0	0	0	1
1.50	0	0	0	0	0	0	0	1
1.51	0	0	0	0	0	0	0	1
1.52	0	0	0	0	0	0	0	1
1.53	0	0	0	0	0	0	0	1
1.54	0	0	0	0	0	0	0	1
1.55	0	0	0	0	0	0	0	1
1.56	0	0	0	0	0	0	0	1
1.57	0	0	0	0	0	0	0	1
1.58	0	0	0	0	0	0	0	1
1.59	0	0	0	0	0	0	0	1
1.60	0	0	0	0	0	0	0	1
1.61	0	0	0	0	0	0	0	1
1.62	0	0	0	0	0	0	0	1
1.63	0	0	0	0	0	0	0	1
1.64	0	0	0	0	0	0	0	1
1.65	0	0	0	0	0	0	0	1
1.66	0	0	0	0	0	0	0	1
1.67	0	0	0	0	0	0	0	1
1.68	0	0	0	0	0	0	0	1
1.69	0	0	0	0	0	0	0	1
1.70	0	0	0	0	0	0	0	1
1.71	0	0	0	0	0	0	0	1
1.72	0	0	0	0	0	0	0	1
1.73	0	0	0	0	0	0	0	1
1.74	0	0	0	0	0	0	0	1
1.75	0	0	0	0	0	0	0	1
1.76	0	0	0	0	0	0	0	1

1.77	0	0	0	0	0	0	0	1
1.78	0	0	0	0	0	0	0	1
1.79	0	0	0	0	0	0	0	1
1.80	0	0	0	0	0	0	0	1
1.81	0	0	0	0	0	0	0	1
1.82	0	0	0	0	0	0	0	1
1.83	0	0	0	0	0	0	0	1
1.84	0	0	0	0	0	0	0	1
1.85	0	0	0	0	0	0	0	1
1.86	0	0	0	0	0	0	0	1
1.87	0	0	0	0	0	0	0	1
1.88	0	0	0	0	0	0	0	1
1.89	0	0	0	0	0	0	0	1
1.90	0	0	0	0	0	0	0	1
1.91	0	0	0	0	0	0	0	1
1.92	0	0	0	0	0	0	0	1
1.93	0	0	0	0	0	0	0	1
1.94	0	0	0	0	0	0	0	1
1.95	0	0	0	0	0	0	0	1
1.96	0	0	0	0	0	0	0	1
1.97	0	0	0	0	0	0	0	1
1.98	0	0	0	0	0	0	0	1
1.99	0	0	0	0	0	0	0	1
1.100	0	0	0	0	0	0	0	1
1.101	0	0	0	0	0	0	0	1
1.102	0	0	0	0	0	0	0	1
1.103	0	0	0	0	0	0	0	1
1.104	0	0	0	0	0	0	0	1
1.105	0	0	0	0	0	0	0	1
1.106	0	0	0	0	0	0	0	1
1.107	0	0	0	0	0	0	0	1
1.108	0	0	0	0	0	0	0	1
1.109	0	0	0	0	0	0	0	1
1.110	0	0	0	0	0	0	0	1
1.111	0	0	0	0	0	0	0	1
1.112	0	0	0	0	0	0	0	1
1.113	0	0	0	0	0	0	0	1
1.114	0	0	0	0	0	0	0	1
1.115	0	0	0	0	0	0	0	1
1.116	0	0	0	0	0	0	0	1
1.117	0	0	0	0	0	0	0	1

1.118	0	0	0	0	0	0	0	1
1.119	0	0	0	0	0	0	0	1
1.120	0	0	0	0	0	0	0	1
1.121	0	0	0	0	0	0	0	1
1.122	0	0	0	0	0	0	0	1
1.123	0	0	0	0	0	0	0	1
1.124	0	0	0	0	0	0	0	1
1.125	0	0	0	0	0	0	0	1
1.126	0	0	0	0	0	0	0	1
1.127	0	0	0	0	0	0	0	1
1.128	0	0	0	0	0	0	0	1
1.129	0	0	0	0	0	0	0	1
1.130	0	0	0	0	0	0	0	1
1.131	0	0	0	0	0	0	0	1
1.132	0	0	0	0	0	0	0	1
1.133	0	0	0	0	0	0	0	1
1.134	0	0	0	0	0	0	0	1
1.135	0	0	0	0	0	0	0	1
1.136	0	0	0	0	0	0	0	1
1.137	0	0	0	0	0	0	0	1
1.138	0	0	0	0	0	0	0	1
1.139	0	0	0	0	0	0	0	1
1.140	0	0	0	0	0	0	0	1
1.141	0	0	0	0	0	0	0	1
1.142	0	0	0	0	0	0	0	1
1.143	0	0	0	0	0	0	0	1
1.144	0	0	0	0	0	0	0	1
1.145	0	0	0	0	0	0	0	1
1.146	0	0	0	0	0	0	0	1
1.147	0	0	0	0	0	0	0	1
1.148	0	0	0	0	0	0	0	1
1.149	0	0	0	0	0	0	0	1
1.150	0	0	0	0	0	0	0	0
1.151	0	0	0	0	0	0	0	1
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	1
3.4	0	0	0	0	0	0	0	1
3.5	0	0	0	0	0	0	0	1
3.6	0	0	0	0	0	0	0	1

3.7	0	0	0	0	0	0	0	1
3.8	0	0	0	0	0	0	0	0
3.9	0	0	0	0	0	0	0	0
3.10	0	0	0	0	0	0	0	0
3.11	0	0	0	0	0	0	0	0
3.12	0	0	0	0	0	0	0	1
3.13	0	0	0	0	0	0	0	1
3.14	0	0	0	0	0	0	0	1
3.15	0	0	0	0	0	0	0	1
3.16	0	0	0	0	0	0	0	0
3.17	0	0	0	0	0	0	0	1
3.18	0	0	0	0	0	0	0	1
3.19	0	0	0	0	0	0	0	1
3.20	0	0	0	0	0	0	0	1
3.21	0	0	0	0	0	0	0	1
3.22	0	0	0	0	0	0	0	0
3.23	0	0	0	0	0	0	0	1
3.24	0	0	0	0	0	0	0	1
3.25	0	0	0	0	0	0	0	1
3.26	0	0	0	0	0	0	0	1
3.27	0	0	0	0	0	0	0	0
3.28	0	0	0	0	0	0	0	1
3.29	0	0	0	0	0	0	0	1
3.30	0	0	0	0	0	0	0	1
3.31	0	0	0	0	0	0	0	1
3.32	0	0	0	0	0	0	0	1
3.33	0	0	0	0	0	0	0	0
4.1	0	0	0	0	1	0	0	0
4.2	0	0	0	0	1	0	0	0
4.3	0	0	0	0	0	0	0	0
5.1	0	0	0	0	1	0	0	0
5.2	0	0	0	0	0	1	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	0	0	0
6.1	0	0	0	0	1	0	1	0
6.2	0	0	0	0	1	0	0	0

Table A.23: Annotated bug report (*Thunderbird:#403907*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
-------------	-----	-----	----	----	-----	----	-----	------

1.1	0	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	0	0	0	0	0	0	0	0
1.9	0	0	0	0	0	0	0	0
1.10	1	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0
1.12	1	0	0	0	0	0	0	0
1.13	1	0	0	0	0	0	0	0
1.14	0	0	0	0	0	0	0	0
1.15	0	0	0	0	0	0	0	0
1.16	0	0	0	0	0	0	0	0
1.17	0	0	0	0	0	0	0	0
1.18	0	0	0	0	0	0	0	0
1.19	0	0	0	0	0	0	0	1
1.20	0	0	0	0	0	0	0	0
1.21	0	0	0	0	0	0	0	0
1.22	0	0	0	0	0	0	0	0
1.23	0	0	0	0	0	0	0	0
1.24	0	0	0	0	0	0	0	0
1.25	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	1	0	0	0	0	0	0
3.2	0	1	0	0	0	0	0	0
3.3	0	1	0	0	0	0	0	0
3.4	0	1	0	1	0	0	0	0
3.5	0	1	0	1	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	1	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	1	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0
5.6	0	0	1	0	0	0	0	0
5.7	0	0	0	0	0	0	0	0

5.8	0	0	1	0	0	0	0	0
5.9	0	0	0	0	0	0	0	0
5.10	0	0	1	0	0	0	0	0
5.11	0	0	0	0	0	0	0	0
5.12	0	0	0	1	0	0	0	0
5.13	0	0	0	0	0	0	0	0
5.14	0	0	0	1	0	0	0	0
5.15	0	0	0	1	0	0	0	0
5.16	0	0	0	1	0	0	0	0
5.17	0	0	0	1	0	0	0	0
5.18	0	0	0	0	0	0	0	0
5.19	0	0	0	1	0	0	0	0
5.20	0	0	0	1	0	0	0	0
5.21	0	0	0	0	0	0	0	0
5.22	0	0	0	0	0	0	0	0
6.1	0	1	0	0	0	1	0	0
6.2	0	1	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	1	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	1	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
9.1	0	0	0	0	1	0	0	0
10.1	0	0	0	0	0	0	0	0

Table A.24: Annotated bug report (*Thunderbird:#296655*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	1
1.2	0	0	0	0	0	0	0	1
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	1	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0

1.12	1	0	0	0	0	0	0	0
1.13	1	0	0	0	0	0	0	0
1.14	0	0	0	0	0	0	0	0
1.15	1	0	0	0	0	0	0	0
1.16	1	0	0	0	0	0	0	0
1.17	1	0	0	0	0	0	0	0
1.18	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0
2.5	0	0	0	0	0	0	0	0
2.6	0	0	0	0	0	0	0	0
2.7	0	0	0	0	0	0	0	0
2.8	0	0	0	0	0	0	0	0
2.9	0	0	0	0	0	0	0	0
2.10	0	0	0	0	0	0	0	0
2.11	0	0	0	0	0	0	1	0
2.12	0	0	0	0	0	0	1	0
2.13	0	0	0	0	0	0	1	0
3.1	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	0
9.3	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	1
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0
14.2	0	0	0	0	0	0	0	0
15.1	0	0	0	0	0	0	0	0
16.1	0	0	0	0	0	0	0	0
17.1	0	0	0	0	0	0	0	0

18.1	0	0	0	0	0	0	0	0
19.1	0	0	0	0	1	0	0	0
19.2	0	0	0	0	1	0	0	0
19.3	0	0	0	0	0	0	0	0
19.4	0	0	0	0	0	0	0	0
19.5	0	0	0	0	0	0	0	0
19.6	0	0	0	0	0	0	0	0
19.7	0	0	0	0	0	0	0	0
19.8	0	0	0	0	0	0	0	0
19.9	0	0	0	0	0	0	0	0
19.10	0	0	0	0	0	0	0	0
19.11	0	0	0	0	0	0	0	0
19.12	0	0	0	0	0	0	0	0
19.13	0	0	0	0	0	0	0	0
19.14	0	0	0	0	0	0	0	0
19.15	0	0	0	0	0	0	0	0
19.16	0	0	0	0	0	0	0	0
19.17	0	0	0	0	0	0	0	0
19.18	0	0	0	0	0	0	0	0
19.19	0	0	0	0	0	0	0	0
19.20	0	0	0	0	0	0	0	0
19.21	0	0	0	0	0	0	0	0
19.22	0	0	0	0	0	0	0	0
19.23	0	0	0	0	0	0	0	0
19.24	0	0	0	0	0	0	0	0
19.25	0	0	0	0	0	0	0	0
20.1	0	0	0	0	1	0	0	0
20.2	0	0	0	0	0	1	0	0
20.3	0	0	0	0	0	0	0	0
20.4	0	0	0	0	0	0	0	0
21.1	0	0	0	0	1	0	0	0
21.2	0	0	0	0	0	1	0	0
22.1	0	0	0	0	0	0	0	0
23.1	0	0	0	0	0	0	0	0
23.2	0	0	0	0	0	0	1	0
23.3	0	0	0	0	1	0	0	0
23.4	0	0	0	0	0	0	0	0
23.5	0	0	0	0	0	0	0	0

Table A.25: Annotated bug report (*KDE:#153211*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
10.1	0	0	0	0	1	0	0	0
10.2	0	0	0	0	1	0	0	0
10.3	0	0	0	0	0	0	0	1
10.4	0	0	0	0	0	0	1	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	1	0	0
12.2	0	0	0	0	0	0	0	0
12.3	0	0	0	0	0	0	0	0
12.4	0	0	0	0	0	0	0	0
12.5	0	0	0	0	0	0	0	0
12.6	0	0	0	0	0	0	0	0
12.7	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
13.2	0	0	0	0	0	0	0	0
13.3	0	0	0	0	0	0	0	0
13.4	0	0	0	0	0	0	0	0

14.1	0	0	0	0	0	0	0	0
14.2	0	0	0	0	0	0	0	0
14.3	0	0	0	0	0	0	0	0
14.4	0	0	0	0	0	0	0	0
14.5	0	0	0	0	1	0	0	0
14.6	0	0	0	0	1	0	0	0
15.1	0	0	0	0	0	0	0	0
15.2	0	0	0	0	0	0	0	0
16.1	0	0	0	0	0	0	0	0
17.1	0	0	0	0	0	0	0	0
18.1	0	0	0	0	1	0	0	0
18.2	0	0	0	0	0	0	0	0
18.3	0	0	0	0	0	0	0	0
18.4	0	0	0	0	0	0	0	1
18.5	0	0	0	0	0	0	0	1
18.6	0	0	0	0	0	0	0	1
18.7	0	0	0	0	0	0	1	0
19.1	0	0	0	0	0	0	0	0
20.1	0	0	0	0	0	0	0	0
20.2	0	0	0	0	0	0	0	0
21.1	0	0	0	0	0	0	0	0
21.2	0	0	0	0	0	0	0	0
21.3	0	0	0	0	0	0	0	0

Table A.26: Annotated bug report (*KDE:#61263*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	1
1.3	1	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
2.1	0	1	0	1	0	0	0	0
2.2	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	1	0	0	0	0
3.3	0	0	0	0	0	0	0	0

3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	0
3.8	0	0	0	0	0	0	1	0
3.9	0	0	0	0	0	0	0	0
3.10	0	0	0	0	0	0	0	0
3.11	0	0	0	0	0	0	0	0
3.12	0	0	0	0	0	0	0	0
3.13	0	0	0	0	0	0	0	0
3.14	0	0	0	0	0	0	0	0
3.15	0	0	0	0	0	0	0	0
3.16	0	0	0	0	0	0	0	0
3.17	0	0	0	0	0	0	0	0
3.18	0	0	0	0	0	0	0	0
3.19	0	0	0	0	0	0	0	0
3.20	0	0	1	0	0	0	0	0
4.1	0	1	0	0	0	0	0	0
4.2	0	1	0	0	0	0	0	0
4.3	0	1	0	0	0	0	1	0
4.4	0	1	0	0	0	0	0	0
4.5	0	1	0	0	0	0	0	0
4.6	0	1	0	0	0	0	1	0
4.7	0	1	0	0	0	0	0	0
4.8	0	1	0	0	0	0	0	0
4.9	0	1	0	0	0	0	0	0
4.10	0	1	0	0	0	0	0	0
4.11	0	1	0	0	0	0	0	0
4.12	0	1	0	0	0	0	0	0
4.13	0	1	0	0	0	0	0	0
4.14	0	1	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	1	0	0
5.5	0	0	0	0	0	0	0	0
5.6	0	0	0	0	0	0	0	0
5.7	0	0	0	0	0	0	0	0
5.8	0	0	0	0	0	0	0	0
5.9	0	0	0	0	0	0	0	0
5.10	0	0	0	0	0	0	1	0

5.11	0	0	0	0	0	0	0	0
5.12	0	0	0	0	0	0	0	0
5.13	0	0	0	0	0	0	0	0
5.14	0	0	0	0	0	0	0	0
5.15	0	0	0	0	0	0	0	0
5.16	0	0	0	0	0	0	0	0
5.17	0	0	0	0	0	0	0	0
5.18	0	0	0	0	0	0	0	0
5.19	0	0	0	0	0	0	0	0
5.20	0	0	0	0	0	0	0	0
5.21	0	0	0	0	0	0	0	0
5.22	0	0	1	0	0	0	0	0
5.23	0	0	0	0	0	0	0	0
5.24	0	0	1	0	0	0	0	0
5.25	0	0	1	0	0	0	1	0
5.26	0	0	1	0	0	0	0	0
5.27	0	0	1	0	0	0	0	0
5.28	0	0	1	0	0	0	1	0
5.29	0	0	1	0	0	0	0	0
5.30	0	0	1	0	0	0	0	0
5.31	0	0	0	0	0	0	0	0
5.32	0	0	1	0	0	0	0	0
5.33	0	0	1	0	0	0	0	0
5.34	0	0	1	0	0	0	0	0
5.35	0	0	1	0	0	0	0	0
5.36	0	0	0	0	0	0	0	0
5.37	0	0	1	0	0	0	0	0
5.38	0	0	1	0	0	0	0	0
6.1	0	1	0	0	0	0	1	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	1	0	0
7.5	0	0	0	0	0	0	0	0
7.6	0	0	0	0	0	0	0	0
7.7	0	0	0	0	0	0	0	0
7.8	0	0	0	0	0	0	0	0
7.9	0	0	0	0	0	0	0	0
7.10	0	0	0	0	0	0	0	0
7.11	0	0	0	0	0	0	0	1
7.12	0	0	0	0	0	0	0	0

7.13	0	0	0	0	0	0	0	0
7.14	0	0	0	0	0	0	0	0
7.15	0	0	0	0	0	0	0	0
7.16	0	0	0	0	0	0	0	0
7.17	0	0	0	0	0	0	0	0
7.18	0	0	0	0	0	0	0	0
7.19	0	0	0	0	0	0	0	0
7.20	0	0	0	0	0	0	1	0
7.21	0	0	0	0	0	0	0	0
7.22	0	0	0	0	0	0	0	0
7.23	0	0	0	0	0	0	0	0
7.24	0	0	0	0	0	0	0	0
7.25	0	0	0	0	0	0	0	0
7.26	0	0	0	0	0	0	0	0
7.27	0	0	1	0	0	0	1	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	0	0
8.6	0	0	0	0	0	0	0	0
8.7	0	0	0	0	0	0	0	0
8.8	0	0	0	0	0	0	0	0
8.9	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	0	0	0
10.4	0	0	0	0	0	0	0	0
11.1	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	0	0	0
12.2	0	0	0	0	0	0	0	0
12.3	0	0	0	0	0	0	0	0
12.4	0	0	0	0	0	0	0	0
12.5	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
13.2	0	0	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0
15.1	0	0	0	0	1	0	0	0

15.2	0	0	0	0	0	0	1	0
------	---	---	---	---	---	---	---	---

Table A.27: Annotated bug report (*KDE:#188311*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	1	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	1	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	1	0	0
8.1	0	0	0	0	1	0	0	0
8.2	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	1	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	0	0	0
10.4	0	0	0	0	0	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	0	0	0
12.2	0	0	0	0	0	0	0	0
12.3	0	0	0	0	0	0	0	0
12.4	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
13.2	0	0	0	0	0	0	0	0
13.3	0	0	0	0	0	0	0	0

14.1	0	0	0	0	0	0	0	0
14.2	0	0	0	0	0	0	0	0
15.1	0	0	0	0	0	1	0	0
16.1	0	0	0	0	0	0	0	0
17.1	0	0	0	0	0	0	0	0
17.2	0	0	0	0	0	0	0	0
17.3	0	0	0	0	0	0	0	0
18.1	0	0	0	0	0	1	0	0
18.2	0	0	0	0	0	0	0	0
18.3	0	0	0	0	0	0	0	0
18.4	0	0	0	0	0	0	0	0
18.5	0	0	0	0	0	0	0	0
18.6	0	0	0	0	0	0	0	0
18.7	0	0	0	0	0	0	0	0
19.1	0	0	0	0	0	0	0	0
20.1	0	0	0	0	1	0	0	0
21.1	0	0	0	0	1	0	0	0
21.2	0	0	0	0	0	0	0	0
21.3	0	0	0	0	0	1	0	0
22.1	0	0	0	0	0	1	0	0

Table A.28: Annotated bug report (*KDE:#88340*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	1
1.2	0	0	0	0	0	0	0	1
1.3	0	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0
3.1	0	1	0	0	0	0	0	0
3.2	0	1	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	1	0	0	0	0	0
4.3	0	0	1	0	0	0	0	0
4.4	0	1	0	1	0	0	0	0
4.5	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0

5.2	0	0	1	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	1	0	0	0	0
5.5	0	0	0	0	0	0	0	0
5.6	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
8.1	0	0	0	0	1	0	0	0

Table A.29: Annotated bug report (*KDE:#66526*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	1	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0
1.7	0	0	0	0	0	0	0	0
1.8	0	0	0	0	0	0	0	0
1.9	0	0	0	0	0	0	0	0
1.10	0	0	0	0	0	0	0	0
1.11	0	0	0	0	0	0	0	0
1.12	0	0	0	0	0	0	0	0
1.13	0	0	0	0	0	1	0	0
1.14	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	0
3.8	0	0	0	0	0	0	0	0
3.9	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	1	0	0

4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	1
5.6	0	0	0	0	0	0	0	1
5.7	0	0	0	0	0	0	0	1
5.8	0	0	0	0	0	0	0	1
5.9	0	0	0	0	0	0	0	1
5.10	0	0	0	0	0	0	0	0
5.11	0	0	0	0	0	0	0	0
5.12	0	0	0	0	0	0	0	0
5.13	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	0	0	0	0	0	0
6.6	0	0	0	0	0	0	0	0
6.7	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	0
9.3	0	0	0	0	0	0	0	0
9.4	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	0	0	1
10.4	0	0	0	0	0	0	0	1
10.5	0	0	0	0	0	0	0	1
10.6	0	0	0	0	0	0	0	1
10.7	0	0	0	0	0	0	0	0
10.8	0	0	0	0	0	0	0	0
10.9	0	0	0	0	0	0	0	1
10.10	0	0	0	0	0	0	0	1
10.11	0	0	0	0	0	0	0	1
10.12	0	0	0	0	0	0	0	1
10.13	0	0	0	0	0	0	0	1

10.14	0	0	0	0	0	0	0	1
10.15	0	0	0	0	0	0	0	1
10.16	0	0	0	0	0	0	0	1
10.17	0	0	0	0	0	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	0	0	0	0	0	0	0
11.4	0	0	0	0	0	0	0	0
11.5	0	0	0	0	0	0	0	0
12.1	0	0	0	0	0	1	0	0
12.2	0	0	0	0	0	0	0	0
12.3	0	0	0	0	0	0	0	1
12.4	0	0	0	0	0	0	0	1
12.5	0	0	0	0	0	0	0	0
12.6	0	0	0	0	0	0	0	0
12.7	0	0	0	0	0	1	0	0
12.8	0	0	0	0	0	0	0	0
12.9	0	0	0	0	0	1	0	0
12.10	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
13.2	0	0	0	0	0	0	0	0
13.3	0	0	0	0	0	0	0	0
13.4	0	0	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0
14.2	0	0	0	0	0	0	0	0

Table A.30: Annotated bug report (*GIMP:#326962*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	0	0	0
5.5	0	0	0	0	0	0	0	0
5.6	0	0	0	0	0	0	0	0

5.7	0	0	0	0	0	0	0	0
5.8	0	0	0	0	0	0	0	0
5.9	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
8.5	0	0	0	0	0	0	0	0
8.6	0	0	0	0	0	0	0	0
8.7	0	0	0	0	0	0	0	0
8.8	0	0	0	0	0	0	0	0
8.9	0	0	0	0	0	0	0	0
8.10	0	0	0	0	0	0	0	0
8.11	0	0	0	0	0	0	0	0
8.12	0	0	0	0	0	1	0	0
8.13	0	0	0	0	0	1	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	0
9.3	0	0	0	0	0	0	0	0
9.4	0	0	0	0	0	0	0	0
9.5	0	0	0	0	0	0	0	0
9.6	0	0	0	0	0	0	0	0
9.7	0	0	0	0	0	0	0	0
9.8	0	0	0	0	0	0	0	0
9.9	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	1	0	0
10.4	0	0	0	0	0	0	0	0
10.5	0	0	0	0	0	0	0	0
10.6	0	0	0	0	0	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	1	0	0	0

Table A.31: Annotated bug report (*Eclipse:#154119*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	1	1	0	1	0	0	0	0
2.1	0	0	1	0	0	0	0	0
2.2	0	0	0	1	0	0	0	0
2.3	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	1	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
4.1	0	0	1	1	0	0	0	0
4.2	0	0	0	0	0	0	0	0
5.1	0	0	1	0	0	0	0	0
5.2	0	0	0	1	0	0	0	0
5.3	0	0	0	1	0	0	0	0
5.4	0	0	0	0	0	0	0	0
5.5	0	0	0	1	0	0	0	0
5.6	0	0	0	0	0	0	0	0
5.7	0	0	0	0	0	0	0	0
5.8	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	1	0	0	0	0	0
6.3	0	0	0	1	0	0	0	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
8.1	0	1	0	1	0	0	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	0	0	0
8.4	0	0	0	0	0	0	0	0
9.1	0	0	1	0	0	0	0	0
9.2	0	0	0	0	0	1	0	0
9.3	0	0	0	1	0	0	0	0
9.4	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0

11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0

Table A.32: Annotated bug report (*Firefox:#238215*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	1
1.2	0	0	0	0	0	0	0	1
1.3	1	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	0	0	0	0	0	1	0	0
1.10	0	0	0	0	0	0	0	0
1.11	0	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	0
3.8	0	0	0	0	0	0	0	0
3.9	0	0	0	0	0	0	0	0
3.10	0	0	0	0	0	0	0	0
3.11	0	0	0	0	0	0	0	0
3.12	0	0	0	0	0	0	1	0
3.13	0	0	0	0	0	0	1	0
3.14	0	0	0	0	0	0	1	0
3.15	0	0	0	0	0	0	0	1
3.16	0	0	0	0	0	0	0	0
3.17	0	0	0	0	0	0	0	0
3.18	0	0	0	0	0	0	0	0
3.19	0	0	0	0	0	0	0	0
4.1	0	1	0	0	0	0	0	0
4.2	0	1	0	0	0	0	0	0

4.3	0	1	0	0	0	0	1	0
4.4	0	1	0	1	0	0	0	0
4.5	0	0	0	0	0	0	0	0
4.6	0	1	0	0	0	0	0	0
4.7	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	1	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	1	0	0	0	0	0
6.5	0	0	1	0	0	0	0	0
6.6	0	0	0	0	0	0	0	0
6.7	0	0	1	0	0	0	0	0
6.8	0	0	1	0	0	0	0	0
6.9	0	0	0	1	0	0	0	0
6.10	0	0	0	1	0	0	0	0
6.11	0	0	0	0	0	0	1	0
7.1	0	0	0	0	0	0	0	0
7.2	0	0	0	0	0	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	0	0	0
7.5	0	0	0	0	0	0	0	0

Table A.33: Annotated bug report (*KDE:#173341*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	1	0	0	0
1.7	0	0	0	0	0	0	0	0
2.1	0	0	0	0	1	0	0	0
2.2	0	0	0	0	1	0	0	0
2.3	0	0	0	0	1	0	0	0
2.4	0	0	0	0	1	0	0	0
2.5	0	0	0	0	0	0	0	0

3.1	0	0	0	0	1	0	0	0
4.1	0	0	0	0	1	0	0	0
4.2	0	0	0	0	1	0	0	0
4.3	0	0	0	0	1	0	0	0
4.4	0	0	0	0	1	0	0	0
4.5	0	0	0	0	1	0	0	0
4.6	0	0	0	0	0	0	0	0
4.7	0	0	0	0	1	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	1	0
6.1	0	0	0	0	1	0	0	0
6.2	0	0	0	0	1	0	0	0
6.3	0	0	0	0	0	1	0	0
6.4	0	0	0	0	1	0	0	0
6.5	0	0	0	0	1	0	0	0
6.6	0	0	0	0	1	0	0	0
7.1	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	1	0	0
8.2	0	0	0	0	0	0	0	0
8.3	0	0	0	0	0	1	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	0	0	0	0	0	0	0
9.3	0	0	0	0	0	0	0	0
9.4	0	0	0	0	0	0	0	0
9.5	0	0	0	0	0	0	0	0
9.6	0	0	0	0	0	0	0	0
10.1	0	0	0	0	0	1	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	0	0	0
10.4	0	0	0	0	0	0	0	0
10.5	0	0	0	0	0	0	0	0
10.6	0	0	0	0	0	0	0	0
10.7	0	0	0	0	0	0	0	0
10.8	0	0	0	0	0	0	0	0
10.9	0	0	0	0	0	0	0	0
10.10	0	0	0	0	0	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	0	0	0	0	0	0	0
11.4	0	0	0	0	0	0	0	0

12.1	0	0	0	0	0	0	0	0
12.2	0	0	0	0	0	0	0	0
12.3	0	0	0	0	0	0	0	0
12.4	0	0	0	0	0	0	0	0
12.5	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
13.2	0	0	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0
14.2	0	0	0	0	0	0	0	0
15.1	0	0	0	0	1	0	0	0
15.2	0	0	0	0	0	0	0	0
15.3	0	0	0	0	0	0	0	0
15.4	0	0	0	0	0	0	0	0
15.5	0	0	0	0	0	0	0	0
15.6	0	0	0	0	0	0	0	0
15.7	0	0	0	0	0	0	0	0
15.8	0	0	0	0	0	0	0	0
16.1	0	0	0	0	0	0	0	0
16.2	0	0	0	0	0	0	0	0
17.1	0	0	0	0	1	0	0	0
17.2	0	0	0	0	0	0	0	0
17.3	0	0	0	0	0	0	0	0
17.4	0	0	0	0	0	0	0	0
17.5	0	0	0	0	0	0	0	0
17.6	0	0	0	0	0	0	0	0
17.7	0	0	0	0	0	0	0	1
17.8	0	0	0	0	0	0	0	1
17.9	0	0	0	0	0	0	0	1
17.10	0	0	0	0	0	0	0	1
17.11	0	0	0	0	0	0	0	1
17.12	0	0	0	0	0	0	0	1
17.13	0	0	0	0	0	0	1	0

Table A.34: Annotated bug report (*KDE:#155920*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0

1.5	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
4.1	0	1	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	1	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	1	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	1	0	0	0	0	0
6.6	0	1	0	1	0	0	0	0
7.1	0	0	1	0	0	0	0	0
7.2	0	0	0	1	0	0	0	0
7.3	0	0	0	1	0	0	0	0
8.1	0	1	0	0	0	0	0	0
8.2	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
10.1	0	0	1	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	0	0	0
11.1	0	0	0	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	0	0	0	0	0	1	0
12.1	0	0	0	0	0	0	0	0
12.2	0	0	0	0	0	0	0	0
12.3	0	0	0	0	0	0	0	0
13.1	0	0	0	0	0	0	0	0
13.2	0	0	0	0	0	0	0	0
13.3	0	0	0	0	0	0	0	0
13.4	0	0	0	0	0	0	0	0
14.1	0	0	0	0	0	0	0	0
14.2	0	0	0	0	0	0	0	0

15.1	0	0	0	0	0	0	0	0
15.2	0	1	0	0	0	0	0	0
15.3	0	0	0	0	0	0	0	0
15.4	0	0	0	0	0	0	0	0
15.5	0	0	0	0	0	0	0	0
15.6	0	0	0	0	0	0	0	0
15.7	0	1	0	0	0	0	0	0
15.8	0	0	0	0	0	0	0	0
15.9	0	0	0	0	0	0	0	0
16.1	0	0	1	0	0	0	0	0
16.2	0	0	0	0	0	0	0	0
16.3	0	0	0	0	0	0	0	0
16.4	0	0	1	0	0	0	0	0
16.5	0	1	0	0	0	0	0	0
16.6	0	0	1	0	0	0	0	0
16.7	0	1	0	0	0	0	0	0
16.8	0	1	0	0	0	0	0	0
17.1	0	0	0	0	0	0	0	0
17.2	0	0	0	0	0	0	0	0
17.3	0	0	0	0	0	0	0	1
17.4	0	0	0	0	0	0	0	1
17.5	0	0	0	0	0	0	0	1
17.6	0	0	0	0	0	0	0	1
17.7	0	0	0	0	0	0	0	1
17.8	0	0	0	0	0	0	0	1
17.9	0	0	0	0	0	0	0	1
17.10	0	0	0	0	0	0	0	1
17.11	0	1	0	0	0	0	0	0
18.1	0	0	1	0	0	0	0	0
18.2	0	0	0	0	0	0	0	0
18.3	0	0	0	0	0	0	0	0
18.4	0	0	1	0	0	0	0	0
18.5	0	0	1	0	0	0	0	0
18.6	0	0	0	0	0	0	0	0
18.7	0	0	0	0	0	0	0	0
19.1	0	0	1	0	0	0	0	0
19.2	0	0	0	0	0	0	0	0
19.3	0	0	0	0	0	0	0	0
19.4	0	0	0	0	0	0	0	0
19.5	0	0	0	0	0	0	0	0
19.6	0	0	0	0	0	0	0	0

20.1	0	0	0	0	0	0	0	0
20.2	0	0	0	0	0	0	0	0
20.3	0	0	0	0	0	0	0	0
20.4	0	0	0	0	0	0	0	0
20.5	0	0	0	0	0	0	0	0
20.6	0	1	0	1	0	0	0	0
21.1	0	0	0	0	0	0	0	0
21.2	0	0	0	0	0	0	0	0
22.1	0	0	1	0	0	0	0	0
22.2	0	0	0	1	0	0	0	0
22.3	0	0	0	0	0	0	0	0
22.4	0	0	0	1	0	0	0	0
22.5	0	0	0	1	0	0	0	0
23.1	0	0	0	0	0	0	0	0
23.2	0	0	0	0	0	0	0	0
24.1	0	0	0	0	1	0	0	0
24.2	0	0	0	0	0	0	0	0
24.3	0	0	0	0	0	0	0	0

Table A.35: Annotated bug report (*KDE:#164545*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0
1.3	0	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	1	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0
1.12	1	0	0	0	0	0	0	0
1.13	1	0	0	0	0	0	0	0
1.14	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0

4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0
5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
5.4	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
7.1	0	0	0	0	0	0	0	0
8.1	0	0	0	0	0	0	0	0
9.1	0	0	0	0	0	0	0	0
9.2	0	1	0	0	0	0	0	0
9.3	0	0	0	0	0	0	0	0
9.4	0	0	0	0	0	0	0	0
9.5	0	1	0	0	0	0	0	0
10.1	0	0	0	0	0	0	0	0
10.2	0	0	0	0	0	0	0	0
10.3	0	0	0	0	0	0	0	0
10.4	0	0	0	0	0	0	0	0
11.1	0	0	1	0	0	0	0	0
11.2	0	0	0	0	0	0	0	0
11.3	0	0	0	0	0	0	0	0
11.4	0	0	1	0	0	0	0	0
11.5	0	0	0	0	0	0	0	0
11.6	0	0	0	0	0	0	0	0
11.7	0	0	0	0	0	0	0	0
11.8	0	0	0	0	0	0	0	0
12.1	0	0	0	0	1	0	0	0
12.2	0	0	0	0	1	0	0	0
12.3	0	0	0	0	0	0	0	0
12.4	0	0	0	0	0	0	0	0
12.5	0	0	0	0	0	0	0	1
12.6	0	0	0	0	0	0	1	0

Table A.36: Annotated bug report (*KDE:#174533*)

Sentence ID	Des	Org	QT	CW	Res	OT	URL	Code
1.1	0	0	0	0	0	0	0	0
1.2	0	0	0	0	0	0	0	0

1.3	0	0	0	0	0	0	0	0
1.4	1	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
1.6	1	0	0	0	0	0	0	0
1.7	1	0	0	0	0	0	0	0
1.8	1	0	0	0	0	0	0	0
1.9	1	0	0	0	0	0	0	0
1.10	0	0	0	0	0	0	0	0
1.11	1	0	0	0	0	0	0	0
1.12	1	0	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0
2.3	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0
3.1	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0
3.6	0	0	0	0	0	0	0	0
3.7	0	0	0	0	0	0	0	0
3.8	0	0	0	0	0	0	0	0
3.9	0	0	0	0	0	0	0	0
3.10	0	0	0	0	0	0	0	0
3.11	0	0	0	0	0	0	0	0
3.12	0	0	0	0	0	0	0	0
3.13	0	0	0	0	0	0	0	0
3.14	0	0	0	0	0	0	0	0
3.15	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0
4.2	0	0	0	0	0	0	0	0
4.3	0	0	0	0	0	0	0	0
4.4	0	0	0	0	0	0	0	0
4.5	0	0	0	0	0	0	0	0
4.6	0	0	0	0	0	0	0	0
4.7	0	0	0	0	0	0	0	0
4.8	0	0	0	0	0	0	0	0
4.9	0	0	0	0	0	0	0	0
4.10	0	0	0	0	0	0	0	0
4.11	0	0	0	0	0	0	0	0
5.1	0	0	0	0	0	0	0	0

5.2	0	0	0	0	0	0	0	0
5.3	0	0	0	0	0	0	0	0
6.1	0	0	0	0	0	0	0	0
6.2	0	0	0	0	0	0	0	0
6.3	0	0	0	0	0	0	0	0
6.4	0	0	0	0	0	0	0	0
6.5	0	0	0	0	0	0	0	0
6.6	0	0	0	0	0	0	0	0
6.7	0	0	0	0	0	0	0	0
6.8	0	0	0	0	0	0	0	0
7.1	0	0	0	0	1	0	0	0
7.2	0	0	0	0	1	0	0	0
7.3	0	0	0	0	0	0	0	0
7.4	0	0	0	0	0	0	0	1
7.5	0	0	0	0	0	0	0	1
7.6	0	0	0	0	0	0	1	0
